②

Naval Research Laboratory

Washington, DC 20375-5000

NRL Memorandum Report 6645

EINMUF: An HF MUF, FOT, LUF Prediction Program

MARK DAEHLER

Ionospheric Effects Branch
Space Science Division
E.O. Hulburt Center for Space Research

May 18, 1990

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved OMB No 0704-0188

1a REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b RESTRICTIVE MARKINGS
2a SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited.
2b DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) NRL Memorandum Report 6645	5 MONITORING ORGANIZATION REPORT NUMBER(S)
6a NAME OF PERFORMING ORGANIZATION Naval Research Laboratory; 6b OFFICE SYMBOL (If applicable) 4181	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (City, State, and ZIP Code) Washington, DC 20375-5000	7b ADDRESS (City, State, and ZIP Code)
8a. NAME OF FUNDING/SPONSORING ORGANIZATION Electromagnetic Compatibility Analysis Ctr.; 8b OFFICE SYMBOL (If applicable)	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER MIPR 89-CS-8084 of 10 February 1989
8c. ADDRESS (City, State, and ZIP Code) 185 Admiral Cochrane Drive Annapolis, MD 21401	10 SOURCE OF FUNDING NUMBERS: PROGRAM ELEMENT NO; PROJECT NO; TASK NO; WORK UNIT ACCESSION NO

11 TITLE (Include Security Classification)
"EINMUF: AN HF MUF, FOT, LUF PREDICTION PROGRAM"

12. PERSONAL AUTHOR(S)
Mark Daehler

13a TYPE OF REPORT	13b TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15 PAGE COUNT
Final	FROM 3/89 TO 12/89	1990 May 18	52

16 SUPPLEMENTARY NOTATION
[illegible]

17 COSATI CODES			18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	High Frequency (HF); Propagation Prediction; Ionospheric Propagation; Sounder Updating

19 ABSTRACT (Continue on reverse if necessary and identify by block number)

This work describes a FORTRAN algorithm, called Subroutine EINMUF, which predicts the Maximum Usable Frequency (MUF), the Frequency of Optimum Transmission (FOT), and the Lowest Usable Frequency (LUF) on a specified HF communications path. A single call to EINMUF produces up to 24 predictions at hourly intervals starting with a specified day-time group DTG. MOF calculations are made with the NOSC MINIMUF85 MUF-prediction program using the first available sunspot number SSN from the following list: a) an effective SSN derived from current oblique-incidence sounder data; b) a 5-day running average of SSNs derived from solar 10.7 cm flux measurements or direct SSN measurements; c) an effective SSN determined from old sounder data; d) SSN extrapolated from previous SSN measurements by the McNish-Lincoln method; e) SSN derived from a mean-value representation of the 11-year solar sunspot number cycle.

20 DISTRIBUTION/AVAILABILITY OF ABSTRACT [✓ UNCLASSIFIED/UNLIMITED] ☐ SAME AS RPT ☐ DTIC USERS	21 ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
22a NAME OF RESPONSIBLE INDIVIDUAL Dr. Mark Daehler	22b TELEPHONE (Include Area Code) 202/767-2891; 22c OFFICE SYMBOL 4181

DD Form 1473, JUN 86 Previous editions are obsolete SECURITY CLASSIFICATION OF THIS PAGE

S/N 0102-LF-014-6603

CONTENTS

FOREWORD

This work describes a FORTRAN algorithm, called Subroutine EINMUF, which provides a prediction, over a specified HF path, of the Maximum Usable Frequency (MUF), the Frequency of Optimum Transmission (FOT), and the Lowest Usable Frequency (LUF). A single call to EINMUF produces a sequence of up to 24 predictions, the first for a specified day-time group DTG, and each of the others for a time one hour following the time of the previous prediction. MUF calculations are made in any one of a number of methods, depending upon the type and age of solar and HF propagation data available at the time of prediction. The calculational heirarchy starts with the use of near-real-time ionospheric oblique-incidence-sounder data (the Maximum Observable Frequency, or MOF) from the specified HF path (or a nearby one), from which an effective sunspot number is calculated and used as a parameter in the NOSC MINIMUF85 MUF-prediction program. This is the Updating technique, and is the preferred method for MUF prediction. If no sounder data is available, but a recent measurement of solar 10.7-cm flux is available, that value will be converted to an equivalent sunspot number and used with MINIMUF85. Lacking a 10.7-cm flux value, a recent sunspot number measurement will be used. If there is inadequate data for a MUF prediction by the previous methods, but there does exist some sounder data (even though it failed to satisfy the requirements of the first attempt at Updating), a second attempt at Updating using the imperfect sounder data will be made. If this is not possible, then a table will be searched for an applicable sunspot number prediction based on measured sunspot numbers from previous years, extrapolated by the McNish-Lincoln method. In the absence of McNish-Lincoln extrapolated data, a sunspot number is derived from a mean-value representation of the 11-year solar sunspot number cycle.

The algorithm was designed to provide usable propagation predictions without extensive expertise from the operator. Decisions regarding suitability of propagation and solar data are therefore made as transparent as possible to the operator.

This work has been funded by the DoD Electromagnetic Compatibility Analysis Center (ECAC) of Annapolis, MD, under procurement document MIPR 89-CS-8084 of 10 February 1989. The subroutine is intended for incorporation in the Frequency Assignment System - HF (FAS-HF), an [illegible] frequency-management system being developed under contract to ECAC by the IIT Research Institute of Annapolis,MD.

EINMUF: AN HF MUF, FOT, LUF PREDICTION PROGRAM

I. GENERAL DESCRIPTION

Subroutine EINMUF is an algorithm designed primarily to provide the HF frequency manager with the best possible estimate of MUF on a specified path, for times up to 24 hours in advance. The method for calculating MUF depends on the propagation information available. The MUF forecasts are derived from the NOSC MINIMUF85 model, in which solar effects on the ionosphere are incorporated via the sunspot number variable. The sunspot number can be determined in various ways, depending on the type of solar or HF propagation information available. Subroutine EINMUF uses the first sunspot number from the following list for which the required data is available:

1. The effective SSN determined from recent sounder measurements of MOF on an appropriate HF path;

2. The sunspot number SSN derived from a current measured value of solar 10.7-cm flux, as reported by the Space Environment Services Center or other agency;

3. A sunspot number SSN which is a five-day running average of daily sunspot numbers, each determined from the 10.7-cm flux, if available, or from the sunspot number reported by the Space Environment Services Center or other agency;

4. SSN predicted by the McNish-Lincoln extrapolation method;

5. SSN determined from mean 11-year sunspot cycle data.

In addition, subroutine EINMUF delivers values of lowest usable frequency (LUF) and frequency of optimum transmission. Values of LUF are determined by the NOSC QLOF program as published in report NOSC TR1189 (August 1987) by D. B. Sailors and W. K. Moision, with modifications as of July 1989 (private communication). Values of FOT are currently set to 85 percent of the MUF.

The package identified as EINMUF comprises the following FORTRAN subroutines and functions:

NEW PROGRAM ELEMENTS	MINMIMUF PROGRAM ELEMENTS	QLOF PROGRAM ELEMENTS
EINMUF1	MUF85MD	QLOF
EFFSSN	FOF2	ABSORB
NISHLINC	PATH	CH
NISHDAT	RAZGC	CHIPINT
DHOUR	SYGN	SUBSOL
INCDTG	GCRAZ	
MEANSSN		
AVSSN		
TRINTERP		
SSNFILE		

Manuscript approved April 25, 1990.

The first column above lists subroutines written specially for ECAC/IITRI. The second column lists the subroutines and functions which calculate MINIMUF MUF predictions, and the third column lists subroutines and functions which calculate QLOF LUF predictions. Figure 1 shows the paths by which the subroutines are called.

The subprograms have the following functions:

EINMUF is a subroutine in which the available solar information is inspected to determine which method of MUF determination is appropriate. Its calling parameters include information about the path, available information about the ionosphere, and the variables LUF, MUF, and FOT to be returned.

EFFSSN is a subroutine which calculates the effective sunspot number corresponding to a measured value of MOF.

NISHLINC is a subroutine which returns, for a specified month and year, a sunspot number prediction which is an extrapolation of measured sunspot numbers calculated by the McNish-Lincoln method.

NISHDAT.DAT is a data file containing McNish-Lincoln predicted sunspot numbers. These data are included as a separate file so that new values can be inserted as they become available, without having to re-compile the EINMUF program. Data file NISHDAT.DAT can be regenerated with newer data using program NISHDAT.FOR.

DHOUR is a subroutine which translates a day-time group DTG into year, day of year, month, day, hour, and a quantity called REFHOUR, which is the number of hours elapsed since 010000ZJAN85. The variable REFHOUR is used to determine the time interval between two values of DTG.

INCDTG is a subroutine which returns a day-time group DTG2 which is exactly one hour later than a specified day-time group DTG1. This subprogram is used to set up the list of times for which MUF, LUF, FOT values are required.

MEANSSN is a subroutine which, for a specified month and year, returns a sunspot number based on the mean 11-year sunspot cycle, starting with a minimum value in September 1986.

AVSSN is a subroutine which manages a file of sunspot number data and calculates a 5-day average sunspot number from the data for the current and four previous days. The data file SSNFILE.DAT is updated everytime EINMUF is called and a previously-unreported sunspot number is included in the variables SSN, DTG2.

MUF85MD is the NOSC MINIMUF routine MUF85, which uses the associated subprograms FOF2,PATH, RAZGC, SYGN, and GCRAZ, with a few minor modifications made for convenience in calling by EINMUF.

QLOF, with the associated subprograms ABSORB, CH, CHPINT, and SUBSOL are NOSC routines for calculating the Lowest Usable Frequency.

TRINTERP is a subroutine which calculates an effective flux by triangular interpolation of effective fluxes determined by sounder measurements on up to three paths, the midpoints of which are "close" to the midpoint of the path for which the MUF prediction is being made. The FORTRAN listing for the subroutine is included with EINMUF, but it's operation has not yet been implemented.

II. CALLING PARAMETERS

The calling statement for subroutine EINMUF is:

```
CALL EINMUF (TLAT, TLONG, RLAT, RLONG, DTG, NUMHRS, CMFLUX, SSN,
               DTG2, LUF, FOT, MUF, HOUR, METHOD, ERRCOD)
```

Input parameters are:

Variable	Type	Description
TLAT	Real	Transmitter latitude [radians] Limits: -PI/2 =< TLAT =< PI/2
TLONG	Real	Transmitter West longitude [radians] Limits: 0 =< TLONG =< 2*PI
RLAT	Real	Receiver latitude [radians] Limits: -PI/2 =< RLAT =< PI/2
RLONG	Real	Receiver West longitude [radians] Limits: 0 =< RLONG =< 2*PI
DTG	Character*12	Day-time group for which first MUF is being requested Limits: 010000ZJAN85 - 312359ZDEC84 (= 0000Z 1 Jan 1985 - 2359Z 31 Dec 2084)
NUMHRS	Integer	Number of successive hours, starting with DTG, for which MUFs are being requested Limits: 1 =< NUMHRS =< 24
CMFLUX	Real	Current 10.7-cm flux value. If no current value is available, enter CMFLUX=0.0. Limits: 43 =< CMFLUX =< 275
SSN	Real	Most recent measurement of sunspot number If no current value is available, enter SSN=-300.0 Limits: -27.31 =< SSN =< 250

DTG2	Character*12	Day-time group of CMFLUX or SSN measurement Limits: Same as for DTG
METHOD(24)	Character*1	(Optional) A lower-case letter entered in the CALL will force a particular method (if data is available); otherwise successive methods from this list will be used: 'm' => updating (Method 1) 'c' => 10.7 cm flux (Method 2) 's' => Measured SSN (Method 3) 'q' => Updating with imperfect sounder data (Method 4) 'l' => McNish-Lincoln SSN (Method 5) 'e' => SSN from mean SSN cycle (Method 6)

<u>Common Block variables used by EINMUF:</u> Common Block /UPMOF/ UPMOF,UPDTG

Variable	Type	Description
UPMOF(96,3)	Real	Array containing 96 measured MOF values [MHz] for each of 3 paths. UPMOF(1,I) is the most recent value for path I; UPMOF(2,I), UPMOF(3,I), etc., are values for successively earlier times. *Limits:* UPMOF(I) > 0
UPDTG(96,3)	Real	Array containing 96 day-time groups for each of 3 paths, corresponding to the MOF values of array UPMOF(96,3). UPDTG(1,I) is the time of the most recent measurement for path I; UPDTG(2,I), UPDTG(3,I), etc. are DTGs corresponding to successively earlier times. Limits: Same as for DTG

OUTPUT PARAMETERS:

Variable	Type	Description
MUF(24)	Real	First NUMHRS elements contain MUF [Mhz] for successive hours starting with DTG Limits: 2.0 <= MUF(I) <= 50.0
FOT(24)	Real	First NUMHRS elements contain FOT [Mhz] for successive hours starting with DTG Limits: 1.7 <= FOT(I) <= 42.5
LUF(24)	Real	First NUMHRS elements contain LUF [Mhz] for successive hours starting with DTG Limits: 1.5 <= LUF(I) <= 50.0
HOUR(24)	Integer	For first NUMHRS elements, HOUR(I) contains integer hour of the MUF(I) prediction Limits: 0 <= HOUR <= 23

METHOD(24)	Character*1	For first NUMHRS elements, METHOD(I) contains the code letter of method used to obtain prediction MUF(I):
		M => Updating (Method 1)
		C => 10.7 cm flux measurement (Method 2)
		S => SSN measurement (Method 3)
		Q => Updating with imperfect sounder data (Method 4)
		L => McNish-Lincoln SSN prediction (Method 5)
		E => SSN from mean 11-year solar cycle (Method 6)

III. CALCULATIONAL METHODS

A. MUF CALCULATIONS

A1. UPDATING METHOD OF MUF PREDICTION

The Updating method is based on the assumption that the MUF on an HF path for times in the near future can be predicted more accurately by taking into account recent MUF measurements on that path (or on a nearby path) than by using a statistical MUF model alone. The value of this technique has been established in a number of studies. See, for example, Reilly and Daehler (1986), Goodman et al. (1984), and Daehler (1984).

Subroutine EINMUF therefore attempts, as a first choice, to find Updating data (in the COMMON block /UPMOF/) which are sufficiently recent to provide an adequate ionospheric assessment. These quantities are contained in the arrays UPMOF(I,J), UPDTG(I,J) which contain, respectively, the MOF measurement and the day-time group DTG of that measurement for paths J = 1 to 3 and measurements I = 1 to 96. At a rate of one sounder measurement per path every 15 minutes, these arrays permit storage of a full 24-hours accumulation of sounder data for each of the three paths. It is assumed that UPDTG(I,1) is the DTG of the most recent measurement on path I, and that other elements of the array, UPDTG(I,2), UPDTG(I,3), etc., are at successively earlier times.

A substantial part of the Updating process is a determination of which of the available measurements are sufficiently close to the time for which a prediction is required to be useful. The comparison of two times is somewhat complicated because times are specified by day-time group, and the two times may correspond to different days, months, or even years. The occurrence of a leap day in the year may also have an effect. For this reason, each day-time group which has to be compared is converted in EINMUF to a reference time, which is defined as the number of hours since 010000ZJAN85. The conversion from DTG to REFHOUR is a part of Subroutine DHOUR.

Subroutine EINMUF is able to predict MUFs for NUMPTS times, starting with the day-time DTG. For the Kth prediction, for which the reference time is REFHOUR(K), the following values are considered acceptable MUF values for Updated predictions:

1. Average of all UPMOFs whose times fall between REFHOUR(K) 2.1 and REFHOUR(K), if there is at least one value;

2. Average of all UPMOFs whose times fall between REFHOUR(K) - 3.1 and REFHOUR(K), if there is at least one value;

3. Average of all UPMOFs whose times fall between REFHOUR(K) - 4.1 and REFHOUR(K), if there is at least one value;

4. Average of all UPMOFs whose times fall between REFHOUR(K) - 5.1 and REFHOUR(K), if there is at least one value;

5. Average of all UPMOFs whose times fall between REFHOUR(K) - 6.1 and REFHOUR(K), if there is at least one value;

6. Average of all UPMOFs whose times fall between REFHOUR(K) - 7.1 and REFHOUR(K), if there is at least one value;

7. Average of all UPMOFs whose times fall between REFHOUR(K) - 25.1 and REFHOUR(K) - 18.9, if there is at least one value.

These criteria are based on data collected on three paths in eastern USA (Daehler, 1984). In this work, it was shown that MUFs predicted by Updating were more accurate than MINIMUF3.5 predictions if the updating data was obtained 7 hours or less previous to the time for which the predictions were made. The advantage is also realized if the Update prediction is made with data collected between 19 and 24 hours earlier, even though Updating with data aged between roughly 8 and 18 hours gives results less accurate than those of MINIMUF3.5. These conclusions are taken from Figure 2.

If the data cannot satisfy any of the above criteria, the Updating method can't be used, and MUF determination by the next method (use of a measured 10.7 cm flux) is attempted. Assuming that Updating data sufficient to satisfy one of the above conditions <u>does</u> exist, the following steps are then taken:

1. The average value of the updating times (PDHR values) is taken;

2. The effective sunspot number SSNEFF corresponding to the average UPMOF value and the average updating time is calculated;

3. SSNEFF is used in MINIMUF85 to calculate the Updated MUF for the required time.

A2. MEASURED 10.7 CM FLUX METHOD OF MUF PREDICTION

In the absence of adequate Updating data, the MUF is calculated using a measured value of solar 10.7 cm flux, if the EINMUF calling statement contains a non-zero value of the parameter CMFLUX. This quantity expresses the measured value of solar 10.7 cm flux in units of 10^{-22} W m^{-2} Hz^{-1}. For use in MINIMUF85, CMFLUX is converted into an equivalent sunspot number according to the relationship between sunspot number according to the relationship between sunspot number and 10.7 cm flux determined by Steward and Leftin (1972) and recommended by Sailors et al. (1986):

$$\text{CMFLUX} = 63.7 + 0.728 * \text{SSN} + 0.00089 * \text{SSN}^2$$

Solving this equation for SSN yields

$$\text{SSN} = 561.8 * [\text{SQRT}(0.303 + 0.00356 * \text{CMFLUX}) - 0.728].$$

The limits to the variable CMFLUX correspond to the range SSN[-27.31, 250] of sunspot number values which are meaningful for MINIMUF85. The corresponding range of meaningful 10.7 cm flux values is CMFLUX[42.2, 275.1].

Current values of 10.7 cm flux, updated approximately every three hours, can be obtained from the answering telephone service of the Space Environment Services Center, Boulder, Colorado. Phone number is (303) 497-3235.

A3. AVERAGE SUNSPOT NUMBER METHOD OF MUF PREDICTION

If a MUF calculation by the previous methods has been unsuccessful, or if the use of a specific sunspot number is required by specification of 's' in the METHOD calling parameter, then EINMUF calculates the MUF using MINIMUF85 with a five-day running-average sunspot number SSN. Function AVSSN manages a file SSNFILE.DAT of daily sunspot numbers, and uses data from this file to calculate five-day average values. The daily values are determined from measurements of CMFLUX, if available (see section A2); otherwise, they are reported measurements of sunspot number. The average value for a given day is calculated as the average of all the SSN values existing in SSNFILE.DAT for that day and the preceding four days, a total of up to five values. If there are no SSN values in SSNFILE.DAT for the appropriate days, Function AVSSN returns the value -300.0.

A smoothed sunspot number is generally appropriate in frequency-management applications because the decreased fluctuations in the smoothed value are considered more characteristic of the ionospheric propagation medium than the typically large fluctions in the daily sunspot number. There is no definitive information on the best averaging time, although it is typically taken to be three to five days. The value used in AVSSN is five days. The number of days over which SSN is averaged can be easily changed by resetting the value of the variable NUMDAYS in function AVSSN. Details on doing this are contained in the section on Function AVSSN and in the FORTRAN listings.

Whenever subroutine EINMUF is called, the daily sunspot number value SSN2 is determined either from CMFLUX or SSN, and Subroutine AVSSN is called to store the values SSN2, DTG2 in file SSNFILE.DAT. Only one value of SSN is stored for each day. If EINMUF is called with a DTG2 corresponding to a day for which there already is a stored value of sunspot number, the stored value is updated with the new value.

The sunspot number averaging feature is also contained in Function AVSSN(DTG,DTG2,SSN). Function AVSSN is called every time EINMUF is entered, so that any available DTG2,SSN2 will be entered into the data file SSNFILE.DAT, even if the MUFs requested in the call do not require calculation by the parameters CMFLUX or SSN. This procedure assures that any sunspot number data present in any call to EINMUF will be entered into SSNFILE.DAT, and will be available for future calls to EINMUF.

The amount of data stored in the file SSNFILE.DAT is limited to thirty DTG2,SSN data pairs. This was considered to be large enough to handle any forseen exercise, without permitting the file to grow to unlimited size. If this amount of data should prove to be inadequate (or unnecessarily large), the quantity of data saved may of course be increased without difficulty.

A4. UPDATING WITH IMPERFECT SOUNDER DATA

If the first three methods fail for lack of adequate sounder and ionospheric data, it is still possible that there is available sounder data which did not meet the specifications for the Updating attempt, but which can be used in a second Updating attempt. Even though the sounder data may be old, it is reasonable to expect that Updating with this data may yield more reliable results than the methods based on extrapolation of historical SSN mean values.

In this procedure, an effective SSN is calculated from every measured value of MOF in the UPMOF array, regardless of the times at which the measurements were made. These effective fluxes are then averaged, and the resulting value is used in MINIMUF85 to determine the MUF.

A5. MUF PREDICTION BASED ON MCNISH-LINCOLN SSN EXTRAPOLATION

Sunspot numbers predicted by the NcNish-Lincoln method (McNish and Lincoln 1949) are published monthly in the Solar-Geophysical Data Prompt Reports Part I (NOAA, Boulder, Colorado). Values for times up to two or three years in advance are published, although beyond one year the predicted SSN values rapidly approach those of the mean values of the thirteen solar cycles (cycles 7 through 20) used in the extrapolation technique.

McNish-Lincoln sunspot number prediction data are maintained in a separate file named NISHDAT.DAT. This permits data to be revised, as new data become available, simply by revising the small file NISHDAT.DAT, without requiring that the entire system be recompiled and/or linked. The program NISHDAT.FOR is provided as a convenient way to generate the data file. The quantities required are: (a) a list of consecutive predicted values for any number of consecutive months; (b) the number of months; and (c) the month number (JAN = 1, etc.) and year (e.g., 1989) of PRED(1), the first predicted value in the array. Instructions are included in the source listing.

Subroutine NISHLINC(DTG,SSN) returns a sunspot number SSN for the time specified by the day-time group DTG. The subroutine extracts the month and year from DTG and determines if there is a value in the data array PRED (which contains data taken from NISHDAT.DAT) for that month and year. If so, the variable SSN returns that value. If there is no predicted value for that month and year, the returned value is SSN=-999.0.

A6. MUF PREDICTION BASED ON MEAN 11-YEAR SUNSPOT CYCLE

This subroutine is included so that a plausible estimate of SSN, including the effect of the 11-year sunspot cycle variation, will be available to EINMUF even for times in the distant future. The result cannot be considered to have great accuracy because sunspot cycles vary substantially in their maximum SSN values, in their rates of SSN rise and fall, and in the times between maxima and minima. The capability may nevertheless be useful for planning and demonstration purposes.

From tabulated dates of sunspot minima in cycles 1-18 (McNish and Lincoln 1949) the mean time between minima is 11.082 years, or very nearly 133 months. Mean SSN values for cycles 8-20 have been presented in graphical from (e.g., Solar Geophysical Data Prompt Reports, February 1989, Number 534, Part I, page 15). These data have been combined to provide the model used in subroutine MEANSSN. The mean sunspot number values were scaled from the curve in the latter reference. This curve extended for only 128 months, apparently because of the low statistical certainty in determining the mean cycle duration, so the last five values for the 133-month period had to be determined by extrapolation. This results in a discontinuity in our model (the cycle starts with SSN=5 and ends with SSN=9), but this should not be serious because the values are very low and the disparity is not excessive when compared with the overall accuracy of the mean-value approximation. The model is illustrated in Figure 3.

Subroutine MEANSSN references time to September 1986, the time of the most recent sunspot minimum, with successive minima being assumed to occur every 133 months. The sunspot number for any given time is determined by the length of time since the preceding minimum, according to Table 1.

Sunspot numbers in the current sunspot cycle (starting September 1986) have been unusually high, as illustrated in Figure 3. Although the maximum has not yet been reached, it can be estimated from the rate of rise that the maximum will be on the order of twice the mean-cycle maximum. Thus for times in the current cycle (that is, through September 1997) subroutine MEANSSN returns an SSN value which is twice the quantity tabulated in Table 1.

Subroutine MEANSSN(MONTH, MYEAR, SSN) is valid for the months September 1986 through December 2085 (MONTH=9, MYEAR=86 through MONTH=12, MYEAR=185). If the subroutine is called for times outside this range, the value SSN=0.0 is returned. Within the range of validity the value 0.0 never occurs, so testing the returned SSN for a zero value can serve as an out-of-range indication, if desired.

B. FOT CALCULATIONS

The frequency of optimum transmission (FOT) can be defined in several different ways. For many purposes it is adequate to define it as a fraction of the MUF, usually between .75 and .90. In EINMUF, the FOT is taken to be 85% of the MUF calculated by MINIMUF.

Alternatively, the FOT can be defined in terms of the highest frequency at which the signal/noise ratio exceeds a minimum required value for a stated fraction of the time during a month (typically 90%). Such a definition requires that the system performance properties be known, which in turn requires that information about transmitter power, antenna gains, and noise levels be known. The use of this FOT definition in the current implementation of EINMUF was not considered because of the unavailability of this information.

C. LUF CALCULATIONS

The lowest usable frequency (LUF) for an HF circuit is calculated with the NOSC QLOF 2.0 program (Sailors and Moision 1987). The subprograms required along with QLOF are subroutines ABSORB, SUBSOL, and DHOUR and functions CH and CHPINT. Of these, listings for QLOF, ABSORB, CH, and CHPINT were taken from the NOSC QLOF 2.0 reference. Subroutine SUBSOL, which is required for calculation of the sub-solar point coordinates, was obtained from NOSC by private communication. In EINMUF, time is passed to QLOF with the day-time group DTG variable, rather than the array ITIME; thus the NRL subroutine DHOUR is used to decompose DTG into hours, minutes, and seconds, etc.

The QLUF implementation in EINMUF does not include a feature of QLOF 2.0 in which the LUF is adjusted to take into account such system parameters as transmitter power, antenna gains, and required S/N ratio. This adjustment is contained in Subroutine ADJUST (and associated program elements GTABLE, GCRAZ, ANTFAC, SYSDAT, IDANT, and ANTNAM). This omission is due in part to the incomplete documentation in the QLOF 2.0 report, and in part to the fact that inclusion of system parameters was not intended in the EINMUF concept.

D. MINIMUF

The MUF-prediction program MINIMUF was created by NOSC to provide a useful frequency-management tool with small computer demands. It incorporates an ionosphere which models the statistical mean ionosphere at all positions on the earth. MINIMUF has been extremely successful, and has undergone a number of revisions as the data base upon which the model is based has become larger. The most current published version is MINIMUF85 (Sailors et al., 1986), the FORTRAN listing of which bears the name Subroutine MUF85. The associated subroutines and functions FOF2, PATH, RAZGC, and SYGN are taken from the same reference. Subroutine GCRAZ, which is CALLed by Subroutine PATH, was taken from Sailors and Moision (1987).

Subroutine EINMUF incorporates MINIMUF85 with only a few modifications which make it more convenient for use by EINMUF, and three changes recommended by NOSC. The modifications are:

1. A change in the calling parameter list to include the day-time group DTG instead of the 6-element array ITIME;

2. A change in the calling parameter list to make the program return the value of G0, the effective zenith angle, for use by EINMUF in calculating the effective sunspot number;

3. Removal of the logical variable V and of statements which are included if V = .TRUE.; according to NOSC (private communication), the V = .TRUE. condition is used only when MINIMUF85 is being called for ray-tracing routines in the PROPHET system.

4. Implementation of three modifications to Subroutine MUF85 and Function FOF2 recommended by NOSC (private communication). Referenced to the FORTRAN listings in Sailors et al (1986), these are:

a. In MUF85, page 95, between statements "K9=0.0" and "GO TO 140" (the two lines preceeding statement 100), add the statement "A4=1.0";

b. In FOF2, page 98, change the fifth line from the bottom of the page from U=COS(T*T) to U=COS(T+T);

c. In FOF2, page 99, change the fifth statement from the end to read:

```
PLR=(2.5+SSN/50.0+U*(0.5+(1.3+0.002*SSN)*YS4)
   +(1.3+0.005*SSN)*COS(PHI-PI*(1.0+B)))
   *(1.0+0.4*(1.0-V*V))*EXP(-V*YS4)
```

E. TRIANGULAR INTERPOLATION

A communications network established in a region in which sounders are operated on two or more paths can in some cases take advantage of the additional sounder information. It has been shown (Reilly and Daehler, 1986) that MOF predictions of increased accuracy can be produced by Updating with MOFs from three sounder paths whose midpoints form a triangle enclosing the midpoint of the path for which the predictions are being made. The use of multiple sounders for acquiring Updating data decreases statistical uncertainties and allows for ionospheric tilts. It also adds a measure of redundancy, since the prediction algorithm can always fall back to a smaller number of Updating paths if data from one sounder should become unavailable.

Since a capability for triangular interpolation is considered useful for the EINMUF prediction algorithm, provision has been made for its incorporation. Subroutine TRINTERP has been included in the EINMUF source listings, and the appropriate UPMOF and UPDTG data arrays have been included. However, the coding in which EINMUF determines the applicability of multiple-sounder interpolation and calls subroutine TRINTERP has not yet been written.

IV. SUBPROGRAM DETAILS

A. SUBROUTINE EFFSSN, A subroutine for calculating the effective sunspot number.

The calling statement for subroutine EFFSSN is

```
CALL EFFSSN(TLAT, TLONG, RLAT, RLONG, DTG, OBSMUF, SSNEFF)
```

Given a path (specified by transmitter coordinates TLAT, TLONG and receiver coordinates RLAT, RLONG), the month, year, and time of day (specified by DTG), and OBSMUF (a measured value of MUF for those conditions), subroutine EFFSSN returns an effective sunspot number SSNEFF. The effective sunspot number SSNEFF is the sunspot number which makes MINIMUF85 produce the MUF value OBSMUF under the stated path and time conditions.

The MUF predicted by the MINIMUF85 model for the given path is proportional to a factor G(SSN,G0):

$$G(SSN,G0) = (\mathrm{SQRT}(6 + [.814\ SSN + 22.23]\ \mathrm{SQRT}(G0)))\ (1.3022 - .00156\ SSN)$$

(Equation 1)

The first factor on the right-hand side represents the dependence of the critical frequency foF2 on sunspot number, as determined by fitting the constants of the factor to vertical-incidence sounder data. (The factor G0 is the cosine of the effective zenith angle, the angle between the vertical and the direction of the sun.) The second factor, a part of the "obliquity"- or "M"-factor, represents the saturation of MUF for very large SSN values. Combined, these two factors describe a non-linear dependence of the MUF on sunspot number. This representation differs from that of MINIMUF 3.5, in which the linear dependence made calculation of effective sunspot number very simple.

Using Equation 1, the relationship between a measured value OBSMUF (and the corresponding effective SSNEFF) and the quantity MUF(SSN) corresponding to an arbitrary sunspot number SSN is

$$R = \frac{OBSMUF}{MUF(SSN)} = \frac{(1.3022-.00156\ SSNEFF)\ (\mathrm{SQRT}(6+[.814\ SSNEFF +22.23]\ \mathrm{SQRT}(G0)))}{(1.3022-.00156\ SSN)\ (\mathrm{SQRT}(6+[.814\ SSN + 22.23]\ \mathrm{SQRT}(G0)))}$$

(Equation 2)

After substituting any value of SSN and calculating MUF(SSN) from MINIMUF85, the above equation can be solved for the effective sunspot number SSNEFF. Before calculating EFFSSN, however, the limits to OBSMUF and SSNEFF will be discussed.

Figure 4 illustrates the quantity G(SSN,G0) as a function of SSN for various values of G0. For typical conditions of SSN between 0 and 250, and G0 greater than 0.1, the function G is a generally monitonically increasing function of SSN. Outside of the typical conditions, however, G doesn't behave the same way. For large values of SSN, G becomes a decreasing function of SSN, which is physically unrealistic. This means that the MUF is double-valued, in that two values of SSN can result in the same MUF value. The effect is more pronounced for lower values of G0; for G0 = .0001, function G is a monitonically decreasing function of SSN over the entire range of SSN between 0 and 250. In addition, very low values of SSN cause the function G0 to become negative, which should imply a negative MUF value. This is again a physically unrealistic condition. While the behavior of the function G over non-typical regions of its parameters is not a hinderance in application of the MINIMUF model, care must be taken in calculating the effective flux corresponding to a measured MUF value, since it can easily happen that extreme ionospheric conditions will yield MUF values well outside the normal range. It is therefore necessary to define limits to the range of the MUF used for updating, and thereby the limits to the effective flux which will be returned.

The value of sunspot number which gives the maximum value of MUF can be found by differentiating Equation 1 and setting the result equal to zero. The resulting extreme value (which turns out indeed to be a maximum) is

SSNHI = 260.04 - 4.914 / SQRT(G0) (Equation 3)

To calculate SSNHI for a specified path, we first adopt a sunspot number value SSNTST (we use the value SSNTST=100) and calculate from MINIMUF85 the corresponding value CMUFTST = MUF(SSNTST). At the same time, the value G0 for the path is calculated. Then SSNHI is calculated from Equation 3, and the MUF corresponding to SSNHI, called CMUFHI, is calculated from Equation 1:

CMUFHI = MUFTST (G(SSNHI,G0) / G(SSNTST,G0)).

It is also appropriate to specify a minimum value of effective sunspot number which will be returned by the subroutine. This value has arbitrarily been set to SSNLO=-27.31, a sunspot number which happens to result in a value for G(SSN,G0) which is independent of G0: for SSN= -27.31 and any value of G0, G = 3.294. (There is no objection to a negative value of effective SSN, since the effective SSN is not a physical reality, but simply a parameter by which MUFs are calculated.) This minimum value of effective sunspot number should be low enough to provide an adequate range in effective sunspot number even under extreme solar conditions, and there is no possibility of the effective flux being so low that the quantity

6 + [.814 SSN + 22.23] SQRT(G0)

in Equation (1) would become negative, resulting in an imaginary value for G.

The minimum sunspot number SSNLO corresponds to a lower MUF limit, called CMUFLO, which can be calculated from Equation 1:

$$\frac{\text{CMUFLO}}{\text{CMUFTST}} = \frac{\text{G(SSNLO,G0)}}{\text{G(SSNTST,G0)}} = \frac{\text{G (-27.31,G0)}}{\text{G(100,G0)}} = \frac{3.294}{\text{G(100,G0)}} .$$

The factor G(100,G0) is calculated from Equation 1 and the value G0 returned when calculating CMUFTST:

```
G(100,G0)  =  1.146 * SQRT(6.0 + 103.6*SQRT(G0)).
```

When calculating an effective flux corresponding to the value OBSMUF, first the value SSNHI (the SSN value which gives the maximum MUF, called CMUFHI) is calculated. In the (unusual) case that SSNHI =< -27.31, there is no meaningful range of SSN over which MINIMUF yields a monitonically-increasing MUF value, and it is not possible to determine an effective flux for the value OBSMUF. In this case the subroutine returns a value SSNEFF = -100.0, in response to which Subroutine EINMUF rejects Updating for this MUF determination and proceeds to the next method.

If SSNHI > -27.31, then Updating can be used for sunspot numbers in the range -27.31 < SSN =< SSNHI, corresponding to MUF values in the range CMUFLO < MUF <= CMUFHI. If OBSMUF >= CMUFHI, then Subroutine EFFSSN returns an effective sunspot number EFFSSN = SSNHI. If OBSMUF <= -27.31, then Subroutine EFFSSN returns an effective sunspot number EFFSSN= SSNHI.

Assuming that the measured value OBSMUF is within the range [MUFLO,MUFHI] (the usual case), then SSNEFF is calculated according to the following procedure:

Using the substitutions

$A = 1.3022$
$B = -0.00156$
$C = 6.0$
$D = 0.814$
$E = 22.23$
$F = \text{SQRT(G0)}$
$G = (A + B\ \text{SSNTST})\ ((\text{SQRT}(C + DF\ \text{SSNLO} + EF))\ R$
$H = C + EF$

$A_1 = (B^2H + 2\ ABDF)/(B^2DF)$
$A_2 = (2\ ABH + A^2DF)/(B^2DF)$
$A_3 = (A^2H - G^2)/(B^2DF)$

equation (2) takes the form

$$\text{SSNEFF}^3 + A_1\ \text{SSNEFF}^2 + A_2\ \text{SSNEFF} + A_3 = 0,$$

which is simply a cubic equation to be solved for SSNEFF. An efficient computer solution to this equation may be sound in standard texts. (For example, see W. H. Press, et al., 1986). The number of roots to the cubic equation is determined from the quantities

$$Q = (A_1^2 - 3\ A_2)\ /\ 9$$

$$R = (2\ A_1^3 - 9\ A_1\ A_2 + 27\ A_3)\ /\ 54\ .$$

If $Q^3 - R^2$ >=0, then there are three real roots; in terms of the quantity T = ARCCOS (R / SQRT(Q^3)), they are

$$X_1 = -2\ \mathrm{SQRT}(Q)\ \mathrm{COS}(\ (T/3) - A_1/3$$

$$X_2 = -2\ \mathrm{SQRT}(Q)\ \mathrm{COS}(\ (T+2\ \mathrm{PI})/3\ ) - A_1/3$$

$$X_3 = -2\ \mathrm{SQRT}(Q)\ \mathrm{COS}(\ (T+4\ \mathrm{PI})/3\ ) - A_1/3.$$

If $Y = R^2 - Q^3 > 0$, then there is only one real root, given by

$$X_4 = -\mathrm{SGN}(R)\ [Y + Q/Y] - A_1/3,$$

where

$$Y = \mathrm{SQRT}(R^2 - Q^3) + |R|^{1/3}.$$

Because the function G(SSN,G0) is continuous and monotonic within the interval SSN = (-27.31, SSNHI), the proper solution to the cubic equation, in the case where there are three real solutions, can be determined simply by looking for a solution lying inside the interval (-27.31, SSNHI). In all situations so far investigated, solution X_1 has yielded the appropriate solution.

B. SUBROUTINE DHOUR, a subroutine for decomposing the day-time group DTG

This subroutine decodes information from the character string DTG (example: 251200ZDEC89) into the following quantities required in EINMUF:

MONTH:	An integer representing the month. 1 =< MONTH =< 12
MYEAR:	An integer representing the year minus 1900. 0 =< MYEAR =< 184
MDOY:	An integer representing the day of year. 1 =< MDOY =< 366

MDAY: An integer representing the day of month. 1 =< MDAY =< 31

MHOUR: An integer representing the integral number of hours.
0 =< MHOUR =< 23

FHOUR: A floating-point number representing the hour.
0.0 =< FHOUR < 24.0

REFHOUR: A floating-point number representing the number of hours elapsed since 010000ZJAN85.

Any legitimate DTG between 1 January 1985 and 31 December 2084 may be decoded with DHOUR. Years-characters (characters 11-12) between 85 and 99 are assumed to correspond to the years between 1985 and 1999, and yield values MYEAR=85 to 99. Years-characters between 00 and 84 are assumed to correspond to the years 2000-2084, and yield values MYEAR=100 to 184.

Subroutine DHOUR contains tests to verify that the DTG passed to it ;is legitimate. If any of these tests fails, the returned valued of REFHOUR is -1. The following tests are performed:

a. DTG must be a string of 12 characters;

b. Character 7 must be an upper or lower case 'Z';

c. Characters 8-10 must correspond to one of the months JAN, FEB, ... DEC. Any combination of upper and lower case letters is permissible.

d. Each of the characters 1-6 and 11-12 must be one of the integers 0,1,...9.

e. Characters 1-2 must evaluate to a number between 1 and the number of days in the month defined by characters 8-10. (Leap year is taken into account.) Characters 3-4 must evaluate to a number between 0 and 23. Characters 5-6 must evaluate to a number between 0 and 59.

C. SUBROUTINE INCDTG, a subroutine for incrementing the day-time group DTG by one hour

Subroutine EINMUF is typically called with a request for a number of MUF values for a series of times separated by one hour, with the first time specified by the day-time group DTG. Since the MUF-calculating routine requires a DTG for each MUF request, the subroutine INCDTG is useful for calculating a series of DTGs, each of which corresponds to a time exactly one hour later than the previous DTG. This seemingly simple process is complicated by the possible change in day, month, or year when moving one hour ahead. An additional complication is involved in accounting for leap years.

D. SUBROUTINE NISHLINC and DATA FILE NISHDAT.DAT, a subprogram and data file to provide access to McNish-Lincoln sunspot number predictions, and Program NISHDAT.FOR, a program for generating the data file NISHDAT.FOR

Forecasts of SSN produced by the McNish-Lincoln method are published monthly in Solar-Geophysical Data Prompt Reports Part I. Current predictions are stored in a data file NISHDAT.DAT; changes can thus be made as they become available, without having to recompile the entire EINMUF (and FAS-HF) routines.

To make generation of the data file NISHDAT.DAT as consistant and reliable as possible, the program NISHDAT.FOR is provided. After program NISHDAT.FOR is edited to include new data, a new NISHDAT.DAT file is generated by running NISHDAT.FOR. The information required for editing NISHDAT.FOR is: (a) the predicted monthly SSN values, assumed to refer to a consecutive series of months; (b) the number of months for which predictions are provided; and (c) the month and year of the first prediction.

E. SUBROUTINE MEANSSN, a subroutine for estimating the sunspot number based on the mean 11-year sunspot cycle.

The monthly mean sunspot numbers for the average solar cycle of 133 months are contained in the 133-element array SMEAN(0:132). Taking Sept 1986 as the start of cycle 22, sunspot numbers are assigned as follows:

Month, Year		SSN
Sept	1986	SMEAN(0)
Oct	1986	SMEAN(1)
.		.
.		.
Jan	1987	SMEAN(4)
.		.
.		.
Jan	1988	SMEAN(16)
.		.
.		.
Sept	1997	SMEAN(132)
Oct	1997	SMEAN(0)
.		.
.		.
Oct	2008	SMEAN(132)
Nov	2008	SMEAN(0)
.		.
.		.
.		.

In subroutine MEANSSN the variable FMUNCE is set to the number of months since Sept 1986, modulo 133, and the corresponding sunspot number, SSN = SMEAN(FMUNCE), is returned.

Since sunspot numbers currently (September 1989) are running about twice as high as the mean values, the MEANSSN values returned for months during the remainder of cycle 22 (through September 1997) are multiplied by a factor of 2.0 to make them more realistic.

F. FUNCTION AVSSN and DATA FILE SSNFILE.DAT

The purpose of Function AVSSN(DTG,DTG2,SSN2) is to return a sunspot number AVSSN which is the running average of sunspot numbers for the (up to) five consecutive days ending with the day of DTG. To store data for previous days, Function AVSSN maintains a data file SSNFILE.DAT, to which new data is added with each call to AVSSN.

The five-day running average is used to provide a smoothed sunspot number which will exhibit fluctuations in amplitude lower than those typical of the daily sunspot number values. While it is generally recognized that use of a smoothed sunspot number is appropriate in frequency-management applications, there is no definitive information on the best averaging time, although it is typically taken to be three to five days. The value used in AVSSN is five days. A different averaging time may be selected by changing the variable NUMDAYS, which is the number of consecutive days immediately preceding DTG2 which will be included in the average. Thus the three-day and five-day averages correspond, respectively, to NUMDAYS values of 2 and 4.

The data file SSNFILE.DAT contains a list of up to thirty DTG2,SSN2 pairs, each pair representing a measurement SSN2 at A time DTG2. Whenever AVSSN is called, the values DTG2,SSN2 are added to file SSNFILE.DAT (assuming SSN2 is not equal to -300). Only one DTG2,SSN2 pair is stored for any given day. If AVSSN is called with a DTG2 corresponding to a day for which data already exists in SSNFILE.DAT, then the new DTG2,SSN2 is assumed to represent more accurate data and is stored in SSNFILE.DAT in place of the existing data for that day. If SSNFILE.DAT is already filled with thirty SSN2 values when AVSSN is called, the data corresponding to the oldest DTG2 is removed, and the new DTG2,SSN2 added in its place. The size of data file SSNFILE.DAT is limited to thirty DTG2,SSN2 pairs so that is should be able to handle any anticipated operation, yet not grow endlessly with continued use.

In calling AVSSN, the value SSN2=-300 is used as a signal that no measured sunspot number is available, and that no change is to be made in SSNFILE.DAT. (The value SSN2=0 is not used for this purpose because it is a legitimate sunspot number value.) Caution should be taken that AVSSN is not called with an undefined value for SSN2, because a zero value will most likely be passed to AVSSN, and that value will be stored in SSNFILE.DAT. The five-day running average is determined by whatever sunspot number values exist in file SSNFILE.DAT for the day DTG2 and the four previous days (but there must be at least one entry). This could result in the use of a single SSN measurement entered four days before the day of DTG2. If the data file is empty and AVSSN is called with SSN2=-300, the value AVSSN=-300 will be returned.

A call to AVSSN requires that the file SSNFILE.DAT be searched for the presence of data for DTGs corresponding to one to four days previous to the day of DTG2. This is facilitated by calculating a reference-day value (REFDAY) for DTG2, which is the number of days between 01 Jan 1985 and the day of DTG2. Similar values REFD(K) are calculated for each DTG in SSNFILE.DAT.

REFERENCES

1. M. Daehler: An HF Communications Frequency-Management Procedure for Forecasting the Frequency of Optimum Transmission. NRL Memorandum Report 5506, 31 December 1984.

2. J. M. Goodman, M. Daehler, M. H. Reilly, and A. J. Martin: A Commentary on the Utilization of Real-Time Channel Evaluation Systems in HF Spectrum Management. NRL Memorandum Report 5454, 28 November 1984.

3. A. G. McNish and J. V. Lincoln: Prediction of Sunspot Numbers. Transactions, American Geophysical Union, Volume 30, Number 5, pp. 673-685, October 1949.

4. W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, Numerical Recipes: The Art of Scientific Computing, Cambridge University Press, p. 146, 1986.

5. Michael H. Reilly and Mark Daehler: Sounder updates for Statistical Model Predictions of Maximum Usable Frequencies on HF Sky Wave Paths. Radio Science, Volume 21, Number 6, Pages 1001-1008, November-December 1986.

6. D. B. Sailors and W. K. Moision: Quiet time lowest observable frequency model uncertainty assessment, NOSC Technical Report TR 1187, August 1987

7. D. B. Sailors, R. A. Sprague, and W. H. Rix: MINIMUF-85: An Improved HF MUF Prediction Algorithm: NOSC Technical Report TR 1121, July 1986.

8. F. G. Stewart and M. Leftin: Relationship between Ottawa 10.7-cm Solar Radio Noise Flux and Zurich Sunspot Number, Telecommunications Journal, Volume 39, pages 159-169, 1972.

```
      SUBROUTINE EINMUF(TLAT,TLONG,RLAT,RLONG,DTG,NUMHRS,CMFLUX,SSNIN,
     A  DTG2, LUF,FOT,MUF,HOUR,METHOD,ERRCOD)
C
C     THIS SUBROUTINE CALCULATES THE MUF WITH THE NOSC MINIMUF 85 PROGRAM
C     USING A SOLAR FLUX INDEX (SUNSPOT NUMBER OR 10.7 CM FLUX) DETERMINED
C     BY THE FIRST AVAILABLE METHOD IN THE FOLLOWING LIST:
C     1) EFFECTIVE SSN DETERMINED FROM RECENT MOF MEASUREMENTS ON A
C        NEARBY SOUNDED PATH;
C     2) CURRENT REPORTED 10.7 CM FLUX;
C     3) CURRENT REPORTED SSN;
C     4) MCNISH-LINCOLN PREDICTED SSN;
C     5) ESTIMATE OF SSN FROM MEAN SUNSPOT CYCLE RECORDS.
C
      REAL LUF(24),FOT(24),MUF(24),UPHOUR(96),CPNT(8),UPMUF(96)
      INTEGER HOUR(24),ERRCOD
      CHARACTER*12 DTG,PDTG,UPDTG,DTG2,OLDPDTG,AVUPDTG,METHOD(24)*1
      COMMON /UPMOF/ UPMOF(96,3),UPDTG(96,3)
      COMMON /GEOD/ CPNT
      DATA DTR/0.017453293/, RTD/57.2957795/
      ERRCOD=0
      SSN=SSNIN
C  Diagnostics are outputted when EINMUF is called with MUF(1)=-100.0
      SING=0.0
      IF (MUF(1) .EQ. -100.0) SING=1.0
C
C
C  If (CMFLUX .NE. 0.0) set SSN2 = sunspot number equivalent of CMFLUX;
C  else if (SSN .NE. -300) set SSN2 = SSN;  else set SSN2=-300.
      SSN2=SSN
      IF (CMFLUX .NE. 0.0)SSN2=561.8*(SQRT(0.303+0.00356*CMFLUX)-0.728)
C
C  Make sure DTG2 is a legitimate day-time group
      CALL DHOUR(DTG2,MYEAR,MDOY,MONTH,MDAY,MHOUR,FHOUR,REFHOUR)
      IF(REFHOUR .EQ. -1.0) SSN2=-300.0
C
C  Call Function AVSSN to store DTG2,SSN2 (if SSN2 .NE. -300) in file
C  SSNFILE.DAT, and calculate the average of sunspot numbers for days
C  of DTG and the four previous days.  If there is no data for this
C  5-day period, the result AVSSN = -300 is returned.  Replace original
C  SSN value with AVSSN for use in EINMUF.
C           TYPE*,' SSN=',SSN,' DTG=',DTG,' DTG2=',DTG2,' SSN2=',SSN2
      SSN=AVSSN(DTG,DTG2,SSN2)
C           TYPE*,' SSN=',SSN,' DTG=',DTG,' DTG2=',DTG2,' SSN2=',SSN2
C
C
      CALL DHOUR(DTG,MYEAR,MDOY,MONTH,MDAY,MHOUR,FHOUR,REFHOUR)
C  If DTG is illegal (REFHOUR=-1), RETURN with error message
      IF (REFHOUR .EQ. -1.0) THEN
         DO 90 K=1,NUMHRS
         LUF(K)=0.0
         FOT(K)=0.0
   90    MUF(K)=0.0
         ERRCOD=1
         GOTO 2000
       END IF
C  Set up the HOURs for which MUFs are requested.
      DO 100 K=1,NUMHRS
        HOUR(K)=MHOUR-1+K
        IF (HOUR(K) .GT. 23) HOUR(K)=HOUR(K)-24
  100   CONTINUE
C  PDTG=present DTG=DTG of 1st MUF request
      PDTG=DTG
C
C
C
C  Diagnostics are outputted when EINMUF is called with MUF(1)=-100.0
      IF (SING .NE. 1.0) GOTO 105
      DO 15 K=1,96
      IF (UPDTG(K,1) .LE. ' ') GOTO 17
      CALL DHOUR(UPDTG(K,1),MYEAR,MDOY,MONTH,MDAY,MHOUR,FHOUR,
     A   UPHOUR(K) )
   15 TYPE 16, K,UPDTG(K,1),UPHOUR(K),UPMOF(K,1)
   16 FORMAT(' K=',I2,'  UPDTG=',A12,'  UPHOUR=',F9.3,'  UPMOF='
     A   F6.2)
   17 TYPE*,' NUMUPS=',K-1
C
C
C
C  Following starts DO loop for the NUMHRS required MUFs
  105 DO 1000 K=1,NUMHRS
C  Determine PDTG for this MUF request (if not the first one)
        IF(K .GT. 1) CALL INCDTG(OLDPDTG,PDTG)
        OLDPDTG=PDTG
C
C  Start Updating procedure
C  Mult. updating MOFs by .75 to make them comparable with MINIMUF MUFs
C  Determine reference hour (UPHOUR) for each updating datum
      DO 110 LK=1,96
        IF (UPDTG(LK,1).LE.' ') GOTO 111
        CALL DHOUR(UPDTG(LK,1),MYEAR,MDOY,MONTH,MDAY,MHOUR,FHOUR
     A  ,UPHOUR(LK))
C  110   UPMUF(LK)=0.75*UPMOF(LK,1)
  110   UPMUF(LK)=UPMOF(LK,1)
C       FACTOR OF 0.75 REMOVED 15 FEB 90, BY MD.
  111 NUMUPS=LK-1
C
C  If directed by METHOD value, start with requested method
      IF (METHOD(K) .EQ. 'm') GOTO 301
      IF (METHOD(K) .EQ. 'c') GOTO 401
      IF (METHOD(K) .EQ. 's') GOTO 501
      IF (METHOD(K) .EQ. 'l') GOTO 601
      IF (METHOD(K) .EQ. 'e') GOTO 701
      IF (METHOD(K) .EQ. 'q') GOTO 550
C
  301 IF (NUMUPS .EQ. 0) GOTO 401
C  Search for acceptable Updating data
        CALL DHOUR(PDTG,MYEAR,MDOY,MONTH,MDAY,MHOUR,FHOUR,PDHR)
        AVUPMUF=0.0
        TIMEDIF=0.0
        NPTS=0
        DO 220 L1=1,NUMUPS
C  Ignore update value if update time comes before time of requested MUF
C  To permit Updating with UPMOF measurements made AFTER the time for
C  which the MUF is being predicted, remove statement 205.
  205       IF (PDHR-UPHOUR(L1) .LT. 0.0) GOTO 220
C  Sum measured MUF and time values if time is within limits
C  NPTS is number of data points which meet specs
          IF (PDHR-UPHOUR(L1) .GT. 2.1) GOTO 230
          AVUPMUF=AVUPMUF+UPMUF(L1)
          TIMEDIF=TIMEDIF+PDHR-UPHOUR(L1)
          NPTS=NPTS+1
  220     CONTINUE
  230   CONTINUE
        IF (NPTS .GE. 1) GOTO 399
        DO 240 L2=L1,NUMUPS
          IF (PDHR-UPHOUR(L2) .GT. 3.1) GOTO 250
          AVUPMUF=AVUPMUF+UPMUF(L2)
          TIMEDIF=TIMEDIF+PDHR-UPHOUR(L2)
  240     NPTS=NPTS+1
        GOTO 399
  250   IF (NPTS .GE. 1) GOTO 399
        DO 260 L3=L2,NUMUPS
          IF (PDHR-UPHOUR(L3) .GT. 4.1) GOTO 270
          AVUPMUF=AVUPMUF+UPMUF(L3)
          TIMEDIF=TIMEDIF+PDHR-UPHOUR(L3)
  260     NPTS=NPTS+1
        GOTO 399
  270   IF(NPTS .GE. 1) GOTO 399
        DO 280 L4=L3,NUMUPS
          IF (PDHR-UPHOUR(L4) .GT. 5.1) GOTO 290
```

```
            AVUPMUF=AVUPMUF+UPMUF(L4)
            TIMEDIF=TIMEDIF+PDHR-UPHOUR(L4)
 280       NPTS=NPTS+1
        GOTO 399
 290    IF (NPTS .GE. 1) GOTO 399
        DO 300 L5=L4,NUMUPS
            IF (PDHR-UPHOUR(L5) .GT. 6.1) GOTO 310
            AVUPMUF=AVUPMUF+UPMUF(L5)
            TIMEDIF=TIMEDIF+PDHR-UPHOUR(L5)
 300       NPTS=NPTS+1
        GOTO 399
 310    IF (NPTS .GE. 1) GOTO 399
        DO 320 L6=L5,NUMUPS
            IF (PDHR-UPHOUR(L6) .GT. 7.1) GOTO 330
            AVUPMUF=AVUPMUF+UPMUF(L6)
            TIMEDIF=TIMEDIF+PDHR-UPHOUR(L6)
 320       NPTS=NPTS+1
        GOTO 399
 330    IF (NPTS .GE. 1) GOTO 399
        DO 375 L7=L6,NUMUPS
            IF (PDHR-UPHOUR(L7) .GT. 25.1) GOTO 375
            IF (PDHR-UPHOUR(L7) .LT. 18.9) GOTO 375
            AVUPMUF=AVUPMUF+UPMUF(L7)
            TIMEDIF=TIMEDIF+PDHR-UPHOUR(L7)
            NPTS=NPTS+1
 375       CONTINUE
C
      IF (NPTS .EQ. 0) GOTO 401
C
C  UPDATING CRITERIA MET.  CALCULATE UPDATED MUF
 399  AVUPMUF=AVUPMUF/NPTS
      AVUPHR=FHOUR-TIMEDIF/NPTS
      DO 289 KL=1,2
 289  IF(AVUPHR .LT. 0.0) AVUPHR=AVUPHR+24.0
      IR=AVUPHR
      IM=(AVUPHR-IR)*60.0
C
C  Note:  AVUPDTG (=average DTG of Updating values) is defined
C  by the average HOUR of the updating values and the DAY for which
C  the MUF is being requested.  This avoids the possibility of
C  calculating the MUF in one month with an effective flux calculated
C  for the previous month.
      AVUPDTG=PDTG(1:2)//CHAR(48+IR/10)//CHAR(48+IR-10*(IR/10))
     A    //CHAR(48+IM/10)//CHAR(48+IM-10*(IM/10))//PDTG(7:12)
      OBSMUF=AVUPMUF
      CALL EFFSSN(TLAT,TLONG,RLAT,RLONG,AVUPDTG,OBSMUF,SSNEFF)
C  On return from EFFSSN, GZERO is returned in OBSMUF, as diagnostic
      EFFGZERO=OBSMUF
C
C
C
C  Diagnostics are outputted when EINMUF is called with MUF(1)=-100.0
      IF (SING .NE. 1.0) GOTO 393
          TYPE 371,K,PDTG,PDHR,NPTS,TIMEDIF/NPTS
          TYPE 372, AVUPDTG,AVUPMUF,SSNEFF,EFFGZERO
 371  FORMAT(' K=',I2,'  PDTG=',A12,'  PDHR=',F9.2,'  NPTS=',I2,
     A    '  AVTIMEDIF=',F6.3)
 372  FORMAT('  AVUPDTG=',A12,'  AVUPMUF=',F8.4,' SSNEFF=',F9.3,
     A    ' G0=',F12.9)
 393  CONTINUE
C
C
C
C  If effective SSN couldn't be calculated, don't Update
      IF(SSNEFF .EQ. -100.0) THEN
          ERRCOD=2
          GOTO 401
       END IF
      CALL MUF85MD(TLAT,TLONG,RLAT,RLONG,PDTG,SSNEFF,MUF(K),GZERO)
      METHOD(K)='M'
```

```
  391   GOTO 900
C
C DETERMINE MUF FROM CURRENT REPORTED 10.7-CM FLUX
  401 IF (CMFLUX .EQ. 0.0) GOTO 501
      SSN=561.8*(SQRT(0.303+0.00356*CMFLUX)-0.728)
      IF (SSN .GT. 250) SSN=250
      IF (SSN .LT. -27.31) SSN=-27.31
      CALL MUF85MD(TLAT,TLONG,RLAT,RLONG,PDTG,SSN,MUF(K),GZERO)
      METHOD(K)='C'
      GOTO 900
C
C DETERMINE MUF FROM CURRENT REPORTED SSN
C  Diagnostic outputted when EINMUF is called with MUF(1)=-100.0
  501 IF (NPTS .EQ. 0 .AND. SING .EQ. 1.0) TYPE 379,K,SSN,PDTG
  379 FORMAT( ' K=',I2, '  SSN=',F8.3,' PDTG=',A12)
      IF (SSN .EQ. -300.0) GOTO 550
      IF (SSN .GT. 250.0) SSN=250.0
      IF (SSN .LT. -27.31) SSN=-27.31
      CALL MUF85MD(TLAT,TLONG,RLAT,RLONG,PDTG,SSN,MUF(K),GZERO)
      METHOD(K)='S'
      GOTO 900
C
C  MUF determinations from Updating, CMFLUX, and SSN methods have
C  been unsuccessful.  Now try Updating using any sounder data
C  whatsoever from array UPMUF.  Each UPMUF value will be used to
C  calculate and effective SSN, and the average of these SSNs will
C  used in a call to MINIMUF to calculate the MUF
  550 IF(NUMUPS .EQ. 0) GOTO 601
      AVSSNEFF=0.0
      DO 560 JK=1,NUMUPS
         CALL EFFSSN(TLAT,TLONG,RLAT,RLONG,UPDTG(JK,1),UPMUF(JK),SSNEFF)
  560    AVSSNEFF=AVSSNEFF+SSNEFF
      AVSSNEFF=AVSSNEFF/NUMUPS
C  If effective SSN couldn't be calculated, don't Update
      IF(AVSSNEFF .EQ. -100.0) GOTO 601
      CALL MUF85MD(TLAT,TLONG,RLAT,RLONG,PDTG,AVSSNEFF,MUF(K),GZERO)
      METHOD(K)='Q'
      GOTO 900
C
C DETERMINE MUF FROM MCNISH-LINCOLN PREDICTED SSN
  601 CALL NISHLINC(PDTG,SSN,KERR)
      IF (KERR .NE. 0) GOTO 701
      CALL MUF85MD(TLAT,TLONG,RLAT,RLONG,PDTG,SSN,MUF(K),GZERO)
      METHOD(K)='L'
      GOTO 900
C
C DETERMINE MUF FROM MEAN SUNSPOT CYCLE SSN VALUES
  701 CALL MEANSSN( MONTH, MYEAR, SSNMN)
      CALL MUF85MD(TLAT,TLONG,RLAT,RLONG,PDTG,SSNMN,MUF(K),GZERO)
      METHOD(K)='E'
      GOTO 900
C
C DETERMINE LUF AND FOT
  900 CALL QLOF(CPNT,PDTG,LUF(K))
      LUF(K)=AMIN1(LUF(K),48.0)
      LUF(K)=AMAX1(LUF(K),1.5)
      FOT(K)=0.85*MUF(K)
 1000 CONTINUE
 2000 RETURN
      END
C
C
C
C
C
      SUBROUTINE EFFSSN(TLAT,TLONG,RLAT,RLONG,DTG,OBSMUF,SSNEFF)
C     THIS SUBROUTINE CALCULATES THE EFFECTIVE SUNSPOT NUMBER SSNEFF
C     WHICH CORRESPONDS TO THE MEASURED MUF VALUE OBSMUF
C
      INTEGER ITIME(6)
      CHARACTER*36 MON$, MN$*9, DTG*12
      REAL CMUFTST,GZERO,CPNT(8)
      DATA A/1.3022/,B/-0.00156/,C/6.0/,D/0.814/,E/22.23/,
     A   MN$/'123456789'/,MON$/'JANFEBMARAPRMAYJUNJULAUGSEPOCTNOVDEC'/,
     B   PI/3.1415926536/
      SSNTST=100.0
      CALL MUF85MD(TLAT,TLONG,RLAT,RLONG,DTG,SSNTST,CMUFTST,GZERO)
      F=SQRT(GZERO)
      H=C+E*F
      RATIO=OBSMUF/CMUFTST
      GG=(A+B*SSNTST)*SQRT(C+(D*SSNTST+E)*F)
      G=RATIO*GG
      CMUFLO=(3.294*CMUFTST)/GG
C  Calculate maximum SSN and MUF allowed by MINIMUF85 for this path
      SSNHI=260.0415-4.914/SQRT(GZERO)
      CMUFHI=CMUFTST*(A+B*SSNHI)*(SQRT(C+(D*SSNHI+E)*SQRT(GZERO)))/GG
C
C  SSNEFF can't be calculated if SSNHI<=-27.31
       IF (SSNHI .LE. -27.31) THEN
          SSNEFF=-100.0
          GOTO 100
         END IF
C
C  Limit SSNEFF to the range [-27.31,SSNHI]
       IF (OBSMUF.GE.CMUFHI) THEN
          SSNEFF=SSNHI
          GO TO 100
         ELSE IF (OBSMUF .LE. CMUFLO)THEN
          SSNEFF=-27.31
          GO TO 100
         END IF
C  Solve cubic equation for SSNEFF
       A1=H/(F*D)+2.0*A/B
       A2=(2.0*A*H)/(B*D*F)+(A*A)/(B*B)
       A3=(A*A*H-G*G)/(B*B*D*F)
       QQ=(A1*A1-3.0*A2)/9.0
       RR=2.0*(A1/54.0)*A1*A1-9.0*A1*(A2/54.0)+27.0*(A3/54.0)
       IF (QQ*QQ*QQ-RR*RR.GE. 0.0) THEN
          NUMROOTS=3
          THETA=ACOS(RR/( (SQRT(QQ))**3 ))
          QIE=SQRT(QQ)
          X1=-2.0*QIE*COS(THETA/3.0)-A1/3.0
          X2=-2.0*QIE*COS((THETA+2.0*PI)/3.0)-A1/3.0
          X3=-2.0*QIE*COS((THETA+4.0*PI)/3.0)-A1/3.0
         ELSE
          NUMROOTS=1
          X1=(SQRT(RR*RR-QQ*QQ*QQ)+ABS(RR))**(1.0/3.0)
     A          +QQ/(SQRT(RR*RR-QQ*QQ*QQ)+ABS(RR))**(1.0/3.0)
          X1=SIGN(1.0,RR)*X1-A1/3.0
         END IF
       SSNEFF=X1
C  GZERO value is returned in variable OBSMUF (a diagnostic)
   100 OBSMUF=GZERO
       RETURN
       END
C
C
C
       SUBROUTINE NISHLINC(DTG,SSN,KERR)
C   This subroutine accepts day-time group DTG and returns a sunspot
C   number prediction for the month and year corresponding to DTG as
C   extrapolated from recent data by the McNISH-Lincoln method.
C   Data are taken from the file NISHDAT.DAT.  If file NISHDAT.DAT
C   doesn't contain an entry for the spcified DTG, the value
C   SSN=-999.0 is returned.
C
       CHARACTER*12,DTG,MN$*36
       INTEGER PRED(100),DAT1MONTH,DAT1YEAR,NUMMONTHS
       OPEN(UNIT=1,FILE='NISHDAT.DAT',STATUS='OLD')
       READ(1,200)NUMMONTHS,DAT1MONTH,DAT1YEAR
```

```
  200 FORMAT(10I5)
      KERR=0
      READ(1,200)(PRED(J),J=1,NUMMONTHS)
      CLOSE(UNIT=1)
      DATA MN$/'123456789'/
      CALL DHOUR(DTG,MYEAR,MDOY,MONTH,MDAY,MHOUR,FHOUR,REFHOUR)
      YEAR=10*INDEX(MN$,DTG(11:11))+INDEX(MN$,DTG(12:12))+1900
      IF (YEAR .LT. 1985) YEAR=YEAR+100
C   MONTHDIFF= time interval in months between 1st data month and
C   month for which SSN is required.
      MONTHDIFF=12*(YEAR-DAT1YEAR)+MONTH-DAT1MONTH
      IF(MONTHDIFF .LT. 0.0 .OR. MONTHDIFF .GT. NUMMONTHS-1) THEN
         KERR=2
      ELSE
         SSN=PRED(MONTHDIFF+1)
      END IF
      RETURN
      END
C
C
C
       SUBROUTINE DHOUR(DTG,MYEAR,MDOY,MONTH,MDAY,MHOUR,FHOUR,REFHOUR)
C
C  INPUT: DTG (EXAMPLE: 251200ZDEC89).  RETURNS MYEAR (=YEAR-1900),
C  MDOY (=DAY OF YEAR), MONTH, MDAY(=DAY OF MONTH), MHOUR (= INTEGRAL
C  PART OF HOUR), FHOUR(HOUR, INCLUDING DECIMAL PORTION), AND
C  REFHOUR(= # OF HOURS SINCE 010000ZJAN85)
C
      CHARACTER*12 DTG, MON$*36,N$*10
      INTEGER IDYS(13)
C  IDYS(MONTH)=number of days in current year preceding the first
C  day of MONTH, assuming no leap year
      DATA IDYS/0,31,59,90,120,151,181,212,243,273,304,334,365/,
     A   N$/'0123456789'/, MON$/'JANFEBMARAPRMAYJUNJULAUGSEPOCTNOVDEC'/
C  A return with REFHOUR=-1 means an illegal DTG was found
      REFHOUR=-1.0
C  DTG must have 12 characters
      IF (LEN(DTG) .NE. 12) GOTO 100
C  Change alpha characters to upper case
      DO 10 K=7,10
         IF(ICHAR(DTG(K:K)) .GT. 90) DTG=DTG(1:K-1)//
     A        CHAR(ICHAR(DTG(K:K))-32)//DTG(K+1:12)
   10    CONTINUE
C  Make sure 1st 6 and last 2 characters are digits, 0-9
      DO 20 K=1,6
   20 IF (INDEX(N$,DTG(K:K)) .EQ. 0) GOTO 100
      DO 30 K=11,12
   30 IF (INDEX(N$,DTG(K:K)) .EQ. 0) GOTO 100
C  End of check for legal DTG
      MONTH=(INDEX(MON$,DTG(8:10))+2)/3
      MDAY= 10*(INDEX(N$,DTG(1:1))-1)+INDEX(N$,DTG(2:2))-1
      MDOY=IDYS(MONTH)+MDAY
      MHOUR=10*(INDEX(N$,DTG(3:3))-1)+INDEX(N$,DTG(4:4))-1
      MIN=  10*(INDEX(N$,DTG(5:5))-1)+INDEX(N$,DTG(6:6))-1
      FHOUR=MHOUR+MIN/60.0
      MYEAR=10*(INDEX(N$,DTG(11:11))-1)+INDEX(N$,DTG(12:12))-1
C  MYEAR.LT.85 => year is between 2000 and 2084
      IF(MYEAR.LT.85) MYEAR=MYEAR+100
C  Count # of leap hours since 010000ZJAN85
      LEAPHRS=24*((MYEAR-85)/4)
      DAYSINM=IDYS(MONTH+1)-IDYS(MONTH)
      IF(MYEAR-4*(MYEAR/4) .EQ. 0.) THEN
         IF(MONTH .GE. 3) MDOY=MDOY+1
         IF(MONTH .GE. 3) LEAPHRS=LEAPHRS+24
         IF(MONTH .EQ. 2) DAYSINM=29
        END IF
C  Declare DTG illegal if mins, hours, or days are unreal
      IF   (MIN .GT. 59 .OR. FHOUR .GE. 24.0 .OR. MDAY .GT. DAYSINM
     A     .OR. DTG(7:7) .NE. 'Z') GOTO 100
      REFHOUR=8760*(MYEAR-85)+24*(IDYS(MONTH)+MDAY-1)+FHOUR+LEAPHRS
  100 RETURN
      END
C
C
C
      SUBROUTINE INCDTG(DTG1,DTG2)
C
C  Input: DTG1.  Returns DTG2, which is 1 hour later than DTG1.
      CHARACTER*12 DTG1,DTG2,MN$*9,MON$*36
       INTEGER MDYS(12)
      DATA MN$/'123456789'/,MDYS/31,28,31,30,31,30,31,31,30,31,30,31/,
     A    MON$/'JANFEBMARAPRMAYJUNJULAUGSEPOCTNOVDEC'/
      MHOUR=10*INDEX(MN$,DTG1(3:3))+INDEX(MN$,DTG1(4:4))+1
      MDYS(2)=28
       IF (MHOUR .EQ. 24) THEN
          MHOUR=0
          MDAY=10*INDEX(MN$,DTG1(1:1))+INDEX(MN$,DTG1(2:2))+1
          MYEAR=10*INDEX(MN$,DTG1(11:11))+INDEX(MN$,DTG1(12:12))
          MONTH=(INDEX(MON$,DTG1(8:10))+2)/3
           IF (MYEAR-4*(MYEAR/4) .EQ. 0) MDYS(2)=29
           IF (MDAY .GT. MDYS(MONTH)) THEN
             MDAY=1
             MONTH=MONTH+1
              IF (MONTH .EQ. 13) THEN
                 MONTH=1
                 MYEAR=MYEAR+1
                  IF (MYEAR .EQ. 100) MYEAR=0
              END IF
          END IF
          DTG2=CHAR(48+MDAY/10)//CHAR(48+MDAY-10*(MDAY/10))
     A        //CHAR(48+MHOUR/10)//CHAR(48+MHOUR-10*(MHOUR/10))
     B        //DTG1(5:7)//MON$(3*MONTH-2:3*MONTH)
     C        //CHAR(48+MYEAR/10)//CHAR(48+MYEAR-10*(MYEAR/10))
        ELSE
          DTG2=DTG1(1:2)//CHAR(48+MHOUR/10)//CHAR(48+MHOUR-10*
     A    (MHOUR/10))//DTG1(5:12)
       END IF
       RETURN
       END
C
C
C
      SUBROUTINE MEANSSN(MONTH,MYEAR,SSN)
C        Accepts MONTH (=1 to 12) and MYEAR (=YEAR-1900).  Returns SSN
C        based on mean 11-year sunspot cycle with minimum on Sep 1986.
C        Valid for months Sep 1986 through Dec 2085 (MONTH=9,MYEAR=86
C        through MONTH=12,MYEAR=185)
       DIMENSION SMEAN(0:132)
C  Array SMEAN contains the 133 SSN values for the 133-month cycle.
C  Array values go from 0 to 132 to facilitate calculation of months
C  since beginning of cycle modulo 133.
       DATA SMEAN /5,5,5.5,6,6.5,7,8,9,10,11,12,14,16,20,21,24,26.5,31,
     A 34,38,42,45,48,53.5,57,60,63,67.5,70,74.5,78,80,83.5,85.5,88,92,
     B 94,96.5,97.5,100,101,102,103,103.5,104,104.5,104.5,105,105,105,
     C 104.5,104.5,104,104,103.5,103,102,101.5,100.5,99.5,98.5,98,97,
     D 96,95,93,91.5,89,87,85,83,81,78.5,76,74,70.5,68.5,66.5,65,62,60,
     E 59,56.5,55,53,52,51,49,48,47.8,45,44,43,41.5,40,38,37,35.5,34,
     F 33,32,30.5,29.5,28.5,27,26,25,24,23.5,22.5,22,20.5,20,19,18,
     G 17.5,17,16,15,14,13.5,13,12.5,12,11,11,10,10,9.8,9.6,9.4,9.2,9/
       FMONATS=3.0+12.0*(MYEAR-87.0)+MONTH
       SSN=0.0
       IF(FMONATS .LT. 0.0 .OR. FMONATS .GT. 1191.0) GOTO 100
       FMUNCE=AMOD(FMONATS,133.0)
       SSN=SMEAN(FMUNCE)
C  SSNs in current cycle are running about twice the mean values
       IF(FMONATS .LE. 132) SSN=2.0*SSN
  100 RETURN
      END
CC
C
```

```
C
      FUNCTION AVSSN(DTG,DTG2,SSN2)
C
C  This function manages a file SSNFILE.DAT which contains a list of
C  up to 30 day-time groups DTGF(K) and corresponding sunspot numbers
C  SSNF(K) (no more than one pair of values per day) and returns
C  a value which is the average of all SSNFs for the day of DTG and
C  the previous NUMDAYS days.  NUMDAYS is typically either 2 or 4,
C  corresponding respectively to a "3-day average" or a "5-day average"
C  sunspot number.  When called, function AVSSN adds DTG2, SSN2 to file
C  SSNFILE.DAT unless SSN2=-300 (which is a signal that DTG2, SSN2 are
C  not valid data), or unless there are already 30 values stored in
C  SSNFILE.DAT with times more recent than DTG2.  It is assumed that
C  either DTG2, SSN2 represent a valid sounder measurement, or that
C  SSN2=-300.0.
C
C  Function AVSSN calls subroutine DHOUR.
C
      CHARACTER*12 DTG,DTG2,DTGF(30)
      DIMENSION SSNF(30)
      INTEGER REFDAY, REFD2, REFD(30)
      NUMDAYS=4
C
C  Read in SSN data from SSNFILE.DAT
      OPEN(UNIT=2,FILE='SSNFILE.DAT',STATUS='UNKNOWN')
      READ(2,100,END=101)  ((DTGF(J),SSNF(J)),J=1,30)
  100 FORMAT(1X,A12,F8.3)
  101 CONTINUE
C       TYPE 120,(DTGF(KJ),KJ=1,30)
C       TYPE 121,(SSNF(KJ),KJ=1,30)
  120 FORMAT(1X,6A13)
  121 FORMAT(1X,6F13.3)
  150 NUMDTGS=J-1
      IF (NUMDTGS .LT. 0) NUMDTGS=0
C
C  Establish REFDAY=number of days between 1 Jan 1985 and DTG
      CALL DHOUR(DTG,MYEAR,MDOY,MONTH,MDAY,MHOUR,FHOUR,,REFHOUR)
      REFDAY=365*(MYEAR-85)+MDOY+(MYEAR-85)/4
C  Establish REFD(K)=number of days between 1 Jan 1985 and DTGF(K)
      DO 270 K=1,NUMDTGS
          CALL DHOUR(DTGF(K),MYEAR,MDOY,MONTH,MDAY,MHOUR,FHOUR,REFHOUR)
  270     REFD(K)=365*(MYEAR-85)+MDOY+(MYEAR-85)/4
C
C  If inputted SSN2,DTG2 aren't valid, don't enter them into SSNFILE.DAT
       IF (SSN2 .EQ. -300.0) GOTO 595
C
C  Since DTG2, SSN2 represent valid data, store them in SSNFILE.DAT
C
C  Find reference day (REFD2) for DTG2
          CALL DHOUR(DTG2,MYEAR,MDOY,MONTH,MDAY,MHOUR,FHOUR,REFHOUR)
          REFD2=365*(MYEAR-85)+MDOY+(MYEAR-85)/4
C
C  If data file was empty, it now will have one entry
       IF (NUMDTGS .EQ. 0) THEN
          DTGF(1)=DTG2
          SSNF(1)=SSN2
          REFD(1)=REFD2
          NUMDTGS=1
          GOTO 500
        END IF
C
C  SSNFILE.DAT wasn't empty.
C  If SSNFILE.DAT already has a value for current day, change value to
C  curent value, if different from earlier value;  also change DTGF(K),
C  since time of day might be different, even if day is the same.
       DO 285 K=1,NUMDTGS
          IF (REFD2 .EQ. REFD(K)) THEN
             SSNF(K)=SSN2
             DTGF(K)=DTG2
             REFD(K)=REFD2
```

```
            GOTO 500
          END IF
  285 CONTINUE
C
C  SSNFILE.DAT doesn't have an entry for the day of DTG2, SSN2
C
C  If SSNFILE.DAT isn't full, simply add new entry
        IF (NUMDTGS .LT. 30) THEN
           DTGF(NUMDTGS+1)=DTG2
           SSNF(NUMDTGS+1)=SSN2
           REFD(NUMDTGS+1)=REFD2
           NUMDTGS=NUMDTGS+1
           GOTO 500
        END IF
C
C  List was full; find oldest data and replace it with DTG2, SSN2
        MINDAY=10000
        DO 287 K=1,30
        IF (MINDAY .GT. REFD(K)) THEN
           MINDAY=REFD(K)
           KMIN=K
        END IF
  287 CONTINUE
        DTGF(KMIN)=DTG2
        SSNF(KMIN)=SSN2
        REFD(KMIN)=REFD2
C
C  Rewrite SSN data to file SSNFILE.DAT
  500 REWIND 2
        WRITE(2,100) ((DTGF(J),SSNF(J)),J=1,NUMDTGS)
C         TYPE 120,(DTGF(KJ),KJ=1,30)
C         TYPE 121,(SSNF(KJ),KJ=1,30)
C         TYPE 122,(REFD(KJ),KJ=1,30)
  122 FORMAT(1X,10I7)
C             TYPE*,' REFDAY=',REFDAY,'  REFD2=',REFD2,' NUMDTGS=',NUMDTGS
  595 CONTINUE
C
C  If there's no SSN data, RETURN an invalid sunspot number
        AVSSN=-300.0
        IF (NUMDTGS .EQ. 0) GOTO 1000
C
C  Calculate average of SSNs for which the respective DTGs are between
C  zero and NUMDAYS in advance of REFDAY, the reference day
C  corresponding to DTG.
        SUM=0.0
        NPTS=0
        DO 600 K=1,NUMDTGS
        IF (REFDAY-REFD(K) .GE. 0 .AND. REFDAY-REFD(K) .LE. NUMDAYS) THEN
           SUM=SUM+SSNF(K)
           NPTS=NPTS+1
        END IF
  600 CONTINUE
        IF (NPTS .EQ. 0) GOTO 1000
        AVSSN=SUM/NPTS
C
 1000 CLOSE(2,STATUS='KEEP')
C             TYPE 1001,NUMDTGS,KMIN,DTGF(KMIN),MINDAY
 1001 FORMAT(' NUMDTGS=',I3,'  KMIN=',I3,'  DTGF(KMIN)=',A12
     A   '  MINDAY=',I6)
        RETURN
        END
C
C
C
        SUBROUTINE TRINTERP(TLAT,TLON,RLAT,RLON,DTG,UPMOF
     A   ,UPDTG,UPLAT,UPTLON,UPRLAT,UPRLON,TMUF,TSSNEFF)
C
C   This subroutine
C      A) accepts Updating data UPMOF from three sounder paths;
C      B) calculates an effective solar 10.7-cm flux for each sounder
```

```
C          midpath point;
C       C) produces an interpolated effective flux for the
C          communications path (the center of which is assumed to lie
C          inside the triangle defined by the three sounder path
C          midpoints);
C       D) converts the effective flux to an effective sunspot number
C          TSSNEFF for the communications path;
C       E) Uses MINIMUF85 to calculate TMUF, the MUF on the communications
C          path;
C       F) RETURNs both TMUF and TSSNEFF.
C
      DIMENSION UMOF(3),UPTLAT(3),UPTLON(3),UPRLAT(3),UPRLON(3)
     A    ,SPATH(8),CPLAT(3),CPLON(3),FLUX(3)
      CHARACTER*12 DTG, UPDTG(3)

C   Calculate sounder path midpoints
C   CPLAT, CPLON = latitude and west longitude of control path
C   (i. e., sounder path) midpoints
      DO 120 J=1,3
      CALL PATH(UPTLAT(J),UPTLON(J),UPRLAT(J),UPRLON(J),SPATH)
      CPLAT(J)=SPATH(2)
      CPLON(J)=SPATH(3)
  120 CONTINUE
C
C   Calculate effective 10.7-cm fluxes corresponding to sounder
C   MUF measurements
      DO 130 J=1,3
      CALL EFFSSN(UPTLAT(J),UPTLON(J),UPRLAT(J),UPRLON(J),UPDTG,
     A    UMOF(J),SSNEFF)
      FLUX(J)=63.7+0.728*SSNEFF+8.9E-4*SSNEFF*SSNEFF
  130 CONTINUE
C
C   Calculate communications path midpoint
      CALL PATH(TLAT,TLON,RLAT,RLON,SPATH)
      COMMLAT=SPATH(2)
      COMMLON=SPATH(3)
C
C   Calculate the 3 coefficients of the linear interpolation formula
      B=   (FLUX(2)-FLUX(3))*(CPLAT(1)-CPLAT(2))
      B=B-(FLUX(1)-FLUX(2))*(CPLAT(2)-CPLAT(3))
      BDIV=      (CPLON(2)-CPLON(3))*(CPLAT(1)-CPLAT(2))
      BDIV=BDIV-(CPLON(1)-CPLON(2))*(CPLAT(2)-CPLAT(3))
      B=B/BDIV
      A=(FLUX(2)-FLUX(3)-B*(CPLON(2)-CPLON(3)))/(CPLAT(2)-CPLAT(3))
      D=FLUX(1)-A*CPLAT(1)-B*CPLON(1)
C
C   Calculate effective flux for the communications path midpoint, and
C   convert to an effective sunspot number
      EFFLUX=A*COMMLAT+B*COMMLON+D
      TEFFSSN=561.8*(SQRT(0.303+0.00356*EFFLUX)-0.728)
C
C   Calculate MUF for the communications path, uSING eff. sunspot number
      CALL MUF85MD(TLAT,TLON,RLAT,RLON,DTG,TEFFSSN,TMUF,GZERO)
      RETURN
      END



      SUBROUTINE MUF85MD(TLAT,TLON,RLAT,RLON,DTG,SSN,CMUF,GZERO)
C     This Version written as a slightly-condensed Version of MUF85.
C
      INTEGER   ITIME(6)
      REAL      CPNT(8),K5,LO,LMT,K8,K9,M9,MLAT,SN(6),CN(6)
      CHARACTER*12,DTG
       COMMON /SUN/  SLAT,SLON
         COMMON /GEOD/ CPNT
C
      DATA    PI/3.14159265/, TWOPI/6.2831853/, HALFPI/1.57079632/,
     *        DTR/0.017453293/, RTD/57.2957795/, RO/6371./
      DATA    S8/250.0/, FM1/0.728/, FMS/0.52998/
     *         FM2/0.00356/, FM3/63.75/, FM4/0.00178/
C
      CALL DHOUR(DTG,MYEAR,MDOY,ITIME(1),ITIME(2),ITIME(3),HOUR,REFHOUR)
      ITIME(4)=60.0*(HOUR-ITIME(3))
      T5= HOUR
      CMUF = 100.0
      YM = 100.0
C
C A3 = 6th order Fourier expansion in terms of Month
      DO 500 N = 1,6
        ARG=PI*FLOAT(2*N)*ITIME(1)/12.0
        SN(N)=SIN(ARG)
        CN(N)=COS(ARG)
  500 CONTINUE
      A3 = .9925+.011*SN(1)+.087*CN(1)-.043*SN(2)+.003*CN(2)-.013*SN(3)
     A      -.022*CN(3)+.003*SN(4)+.005*SN(5)+.018*CN(6)
C
C A1,A2 are linear functions of SSN
      A1 = .814*SSN+22.23
      A2 = 1.3022-.00156*SSN
C
C # OF HOPS: 1 HOP FOR RANGE <= 4000 KM (.6278 RADIANS), 2 HOPS OTHERWISE
      CALL PATH (TLAT,TLON,RLAT,RLON,CPNT)
      G1 = CPNT(1)
      AZIM = CPNT(8)
      AK6 = 1.59*CPNT(1)
      IF (AK6 .LT. 1.0)AK6=1.0
      K5 = 1.0/AK6
      IF (K5 .NE. 1.0)K5=.5
      KHOP = INT(CPNT(1)/.62784)+1
      KKHOP = KHOP
      IF (CPNT(1) .GT. 0.94174)KKHOP=2*KHOP-1
C
C Loop through all hops
      DO 160 K1 = 1,KKHOP
C
C PL = Range (in radians) to Control Point (CP)
C CP = Midpoint for 1-hop; 2000Km from each end for 2-hops
      IF (KHOP .EQ. 1)PL=G1/2.0
      IF (KHOP .EQ. 2 .AND. K1 .EQ. 1)PL=0.31392
      IF (KHOP .EQ. 2 .AND. K1 .EQ. 2)PL=G1-0.31392
      IF (CPNT(1) .GT. .94174) THEN
        XK1 = K1
        AK1 = XK1/(2.0*KHOP)
       ELSE
        AK1 = 1.0/(2.0*AK6)+FLOAT(K1-1)*(.9999-1.0/AK6)
       END IF
      PL = G1*AK1
C
C LO,WO = Latitude and West Longitude of Control Point
  600 CALL RAZGC(RLAT,RLON,PL,AZIM,LO,WO)
C
C LMT = Local mean time at Control Point, in hours
      IF (WO .GE. 0.0) THEN
        LMT = WO
       ELSE
        LMT = WO+TWOPI
       END IF
      LMT = T5-LMT*RTD/15.0
      IF (LMT .LT. 0.0) THEN
        LMT = LMT+24.0
       ELSE IF (LMT .GE. 24.0) THEN
        LMT = LMT-24.0
       END IF
C
C MLAT = Geomagnetic Latitude at Control Point
      SMG = 0.9792*SIN(LO)+0.2028*COS(LO)*COS(WO-1.2043)
      SMG = AMAX1(AMIN1(SMG,1.0),-1.0)
      MLAT = ASIN(SMG)
C
```

```
C GYRO = Gyro Frequency (=0 for LAT=>55 degrees)
      IF (ABS(MLAT) .LT. 0.95993) THEN
         GYRO = 0.0
        ELSE
         GYRO = 0.3789*SQRT(1.0+3.0*SMG*SMG)-0.5
       END IF
C
C Y1=2*PI*DATE/365.25; Y2=-Solar Declination;  k8=Time of local noon
      Y1 = 0.0172*(10.0+FLOAT(ITIME(1)-1)*30.4+ITIME(2))
      Y2 = 0.409*COS(Y1)
      K8 = 3.82*WO+12.0+0.13*(SIN(Y1)+1.2*SIN(2.0*Y1))
      IF (K8 .GT. 24.0) THEN
         K8 = K8 -24.0
        ELSE IF (K8 .LE. 0.0) THEN
         K8 = K8+24.0
        END IF
C
C   M9 = M-Factor = MUF/FOF2C
      M9 = AMIN1(2.5*G1*K5, HALFPI)
      M9 = SIN(M9)
      M9 = 1.0+2.5*M9*SQRT(M9)
C
C K9=length of daylight;   t=time of sunrise;   t4=time of sunset
      IF (COS(LO+Y2) .GT. -0.26) GOTO 100
C NO DAYLIGHT on PATH at ANY TIME DURING the DAY
        GO = 0.0
        K9 = 0.0
        A4=1.0
        GOTO 140
  100 CONTINUE
C
      K9 = (-0.26+SIN(Y2)*SIN(LO))/(COS(Y2)*COS(LO)+1.0E-3)
      K9 = AMAX1(AMIN1(K9,1.0),-1.0)
      K9 = 12.0-ASIN(K9)*7.6394
      T  = K8-K9/2.0
      IF (T .LT. 0.0)T=T+24.0
      T4 = K8+K9/2.0
      IF (T4 .GT. 24.0)T4=T4-24.0
      CO = ABS(COS(LO+Y2))
      T9 = 9.7*(AMAX1(CO,.1))**9.6
      T9 = AMAX1(T9,0.1)
      T6 = T5
      IF ((T4 .LT. T .AND. (T5-T4)*(T-T5) .GT. 0.0) .OR.
     *      (T4 .GE. T .AND. (T5-T)*(T4-T5) .LE. 0.0)) GOTO 120
C
C   DAY TIME at CONTROL POINT
        IF (T .GT. T5)T6=T6+24.0
C
C   LOCAL TIME CONVERSION
C   T5 is LOCAL TIME; WO is LONGITUDE in RADIANS
      Z = WO*(180.0/3.14159265)
C
C   LOCAL TIME DEPENDENT FACTOR for M-FACTOR
        HRLCL = T5-Z/15.0
        IF (HRLCL .GE. 24.0)HRLCL=HRLCL-24.0
        IF (HRLCL .LT.  0.0)HRLCL=HRLCL+24.0
        A4 = 1.11-.01*HRLCL
        G9 = PI*(T6-T)/K9
        G8 = PI*T9/K9
         U = (T-T6)/T9
         U = AMIN1(AMAX1(U,-87.0),+87.0)
        U1 = -K9/T9
        U1 = AMIN1(AMAX1(U1,-87.0),+87.0)
        GO = CO*(SIN(G9)+G8*(EXP(U)-COS(G9)))/(1.0+G8*G8)
        G3 = CO*(G8*(EXP(U1)+1.0))
     A        *EXP((K9-24.0)/2.0)/(1.0+G8*G8)
        GO = AMAX1(GO,G3)
        GOTO 140
  120 CONTINUE
C
```

```
C   NIGHT TIME at CONTROL PCINT
C
C        6th Order Fourier series night time factor for M-factor,
C            based on hours after sunset, T2
C
          IF (T4 .GT. T5)T6=T6+24.0
          T1 = T6-T4
          T2 = 14.0*T1/(24.0-K9)
          AG = PI*(T2+1.0)/15.0
          AG1 = 2.0*AG
          AG2 = 4.0*AG
          AG3 = 6.0*AG
          AG4 = 8.0*AG
          AG5 = 10.0*AG
          AG6 = 12.0*AG
          A14 = 1.0195-.06*SIN(AG1)-.037*COS(AG1)+.018*SIN(AG2)
          A24 = -.003*COS(AG2)+.025*SIN(AG3)+.018*COS(AG3)
          A34 = .007*SIN(AG4)-.005*COS(AG4)+.006*SIN(AG5)
          A44 = .017*COS(AG5)-.009*SIN(AG6)-.004*COS(AG6)
         ,A4 = A14+A24+A34+A44
C
            G8 = PI*T9/K9
            U = (T4-T6)/2.0
            U = AMIN1(AMAX1(U,-75.0),+75.0)
            U1 = -K9/T9
            U1 = AMIN1(AMAX1(U1,-75.0),+75.0)
            GO = CO*(G8*(EXP(U1)+1.0))*EXP(U)/(1.0+G8*G8)
  140       CONTINUE
C
           G2 = SQRT(6.0+A1*SQRT(GO))+GYRO
           G2 = G2*(1.0-0.1*EXP((K9-24.0)/3.0))
           G2 = G2*(1.0+(1.0-SYGN(TLAT)*SYGN(RLAT))*0.1)
           G2 = G2*(1.0-0.1*(1.0+SYGN(ABS(SIN(LO))
     *            -COS(LO))))
C
       IF (ABS(MLAT) .GE. 0.95993) THEN
C
C   FOF2 corrects for polar region FOF2. The result is G2 if
C        not in polar region
C
          G2 = M9*FOF2(G2,LMT,ITIME,LO,WO,MLAT,SSN)
        ELSE
          G2 = G2*M9
      END IF

         G2 = G2*A2*A3*A4
           CMUF = AMIN1(CMUF,G2)
  160 CONTINUE
      GZERO=GO
      CMUF = AMIN1(AMAX1(CMUF,2.0),50.0)
         SLAT=Y2
         SLON=Y1
C
      RETURN
      END


      FUNCTION FOF2(FF2,LMT,ITIME,LAT,LON,MLAT,SSN)
! * * * * * * * * * * * * * * * FUNCTION FOF2  * * * * * * * * * * * * * * *
!     X = FOF2(FF2,LMT,ITIME,LAT,LON,MLAT,SSN)
!
!     This FUNCTION Corrects the F2-Layer Critical Frequency Computed
!             by MUF35 for Polar Latitude uSING the CHIU Model.
!
!     Reference to be Supplied when AVailable.
!
!     INPUT:
```

```
!         FF2 - Critical Frequency from MUF35 in MHz - REAL
!         LMT - Local Mean Time at LAT, LON in Hours - REAL
!       ITIME - Integer Array containing  Month, Day - INTEGER
!                  Hour, Minute, Julian Day, and Year
!         LAT - Geographic  Latitude  in  Radians    - REAL
!         LON - Geographic  West Longitude in Radians- REAL
!        MLAT - Magnetic  Latitude  in  Radians      - REAL
!         SSN - Sunspot Number                       - REAL
!
!     OUTPUT:
!        FOF2 - The F2-layer Critical Freq. in MHz   - REAL
!
!     CALLed by SUBROUTINE or FUNCTION:  MUF35
!
!     SUBROUTINEs and FUNCTIONs CALLed:  None
!
! *** COMMON BLOCKS Referenced: None
      INTEGER   ITIME(6)
      REAL      LAT,LMT,LON,MLAT,MLON
      DATA  PI/3.1415926/

      PHI = LMT*PI/12.0
      TMO = ITIME(1)+(ITIME(2)+ITIME(3)/24.0+ITIME(4)/1440.0)/30.0-0.5
      CMLAT = COS(MLAT)
      MLON = COS(LAT)*SIN(LON-1.2043)/CMLAT
      MLON = AMAX1(AMIN1(MLON,1.0),-1.0)
      MLON = ASIN(MLON)
       X = (2.2+(0.2+SSN/1000.0)*SIN(MLAT))*CMLAT
      FF = EXP(-(X**6))
      GG = 1.0-FF
       T = PI*TMO/12.0
       V = SIN(T)
      IF (MLAT .GE. 0.0) THEN
          W = EXP(-1.2*(COS(MLAT-0.41015*COS(PHI))-CMLAT))
        PLR = (2.0+0.012*SSN)*W*(1.0+0.3*V)
      ELSE
          U = COS(T+T)
          Y = SIN(MLON/2.0)
         YS = COS(MLON/2.0-PI/20.0)
          Z = SIN(MLON)
         ZA = SQRT(ABS(Z))
         AM = 1.0+V
          B = V*((Y-Z)/2.0-Y**8)-AM*U*(Z/ZA)*EXP(-4.0*Y*Y)
        YS4 = YS**4
        PLR = (2.5+SSN/50.0+U*(0.5+(1.3+0.002*SSN)*YS4)
     *           +(1.3+0.005*SSN)*COS(PHI-Pi*(1.0+B)))
     *           *(1.0+0.4*(1.0-V*V))*EXP(-V*YS4)
      END IF
      FOF2 = 2.85*SQRT(GG*FF2*FF2/8.12+0.66*FF*PLR)
      RETURN
      END



      SUBROUTINE PATH(TLAT,TLON,RLAT,RLON,CPNT)

! * * * * * * * * * * * * * * * SUBROUTINE PATH * * * * * * * * * * * * * * *
!     CALL PATH(TLAT,TLON,RLAT,RLON,CPNT)
!
!     This  Routine Computes the Range,  Azimuth,  and  Control Point
!     Coordinates for a GiVen  Propagation Path.  The  Method Assumes
!     a Spherical Earth with a Radius of 6371 km.  The Required Input
!                          for this Module is:
!        TLAT - Transmitter Latitude in Radians
!        TLON - Transmitter West Longitude in Radians
!        RLAT - ReceiVer Latitude in Radians
!        RLON - ReceiVer West Longitude in Radians
!
!     This SUBROUTINE Returns the Following Information in an 8 word
!                           Real Array (CPNT):
!       CPNT(1) - Distance between the RCVR and XMTR in Radians
!       CPNT(2) - Latitude of Midpoint in Radians
!       CPNT(3) - West Longitude in Radians
!       CPNT(4) - Latitude of Point 1000km from the RCVR in Radians
!       CPNT(5) - West Longitude of Point 1000km from RCVR in Radians
!       CPNT(6) - Latitude of Point 1000km from XMTR in Radians
!       CPNT(7) - West Longitude of Point 1000km from XMTR in Radians
!       CPNT(8) - Azimuth from RCVR to XMTR in Radians
!
!     CPNT(4) through CPNT(7) will not be Computed for Paths less than
!               1000km (0.15696 Radians) in Length.
!
!     SUBROUTINEs and FUNCTIONs Used:  GCRAZ
!                                      RAZGC
! *** COMMON BLOCKS Referenced: None
      DIMENSION CPNT(8)
! *** GET RANGE and AZIMUTH
      CALL GCRAZ(RLAT,RLON,TLAT,TLON,CPNT(1),CPNT(8))

! *** GET MIDPOINT COORDINATES
      PL = CPNT(1)/2.0

      CALL RAZGC(RLAT,RLON,PL,CPNT(8),CPNT(2),CPNT(3))

! *** IS the PATH LENGTH >= 1000km?
      IF (CPNT(1) .LT. 0.15696) GOTO 100

! *** YES: GET COORDINATES of 1000km POINTS
        PL = 0.15696
      CALL RAZGC(RLAT,RLON,PL,CPNT(8),CPNT(4),CPNT(5))
        PL = CPNT(1)-0.15696
      CALL RAZGC(RLAT,RLON,PL,CPNT(8),CPNT(6),CPNT(7))
  100 CONTINUE
      RETURN
      END



      SUBROUTINE RAZGC(LAT1,LON1,RANGE,AZIM,LAT2,LON2)
! * * * * * * * * * * * * * * SUBROUTINE RAZGC * * * * * * * * * * * * * *
!     CALL RAZGC(LAT1,LON1,RANGE,AZIM,LAT2,LON2)
!
!     This Routine Computes the Latitude and West Longitude (LAT2,LON2)
!     of a Point, a  Specified Range from a Given Point  on the Earth's
!     Surface. Also Required for Input is the Azimuth (AZIM) to the New
!     Point  in Radians.  This  Method  Assumes a Spherical  Earth  and
!     Recognizes  the Degenerate Cases of the Given  Point being at the
!     North or South pole.  For the Degenerate Cases, Azimuth Should be
!     0,  or PI and LON2 is Undefined.  HoweVer, Azimuth is not Checked
!     and  LON2 is Arbitrarily Set Equal  to LON1.  This Routine Recog-
!     nizes the Degenerate Case when Range is set to 0. All Coordinates
!                              are in Radians.
!
!     SUBROUTINEs and FUNCTIONs Used:  None
!
! *** COMMON BLOCKS Referenced: None
      REAL  LAT1,LON1,LAT2,LON2
      DATA  PI/3.14159/,TWOPI/6.28318/,HALFPI/1.570796/
      DATA  RTD/57.295779/,DTR/0.017453/
! *** TEST for DEGENERATE CASES
      IF (ABS(LAT1-HALFPI) .GT. 1.0E-5) GOTO 100

! *** THE GIVEN POINT is the NORTH POLE
        LAT2 = HALFPI-RANGE
        LON2 = LON1
        GOTO 200
  100 CONTINUE
```

```
      IF (ABS(LAT1+HALFPI) .GT. 1.0E-5) GOTO 120
! *** THE GIVEN POINT Is the SOUTH POLE
        LAT2 = RANGE-HALFPI
        LON2 = LON1
        GOTO 200
  120 CONTINUE
      IF (RANGE .GT. 0.0) GOTO 130
! *** POINT 2 COINCIDENT with POINT 1
        LAT2 = LAT1
        LON2 = LON1
        GO TO 200
  130 CONTINUE
! *** GENERAL CASE
      S1 = SIN(LAT1)
      C1 = COS(LAT1)
      C2 = COS(RANGE)
      CA = S1*C2+C1*SIN(RANGE)*COS(AZIM)
      CA = AMIN1(AMAX1(CA,-1.0),+1.0)
       A = ACOS(CA)
! *** TEST If DESTINATION ENDS UP on the POLES
      IF (ABS(A) .GT. 1.0E-5) GOTO 140
        LAT2 = HALFPI
        LON2 = LON1
        GOTO 200
  140 CONTINUE
      IF (ABS(A-PI) .GT. 1.0E-5) GOTO 150
        LAT2 = -HALFPI
        LON2 = LON1
        GOTO 200
  150 CONTINUE
! *** EVERYTHING SEEMS O.K. - GET DESTINATION COORDINATES
        CG = (C2-S1*CA)/(C1*SIN(A))
        CG = AMIN1(AMAX1(CG,-1.0),+1.0)
        G = ACOS(CG)
        LAT2 = HALFPI-A
        SA = SIN(AZIM)
        IF (SA .GE. 0.0)LON2=AMOD(LON1-G,TWOPI)
        IF (SA .LT. 0.0)LON2=AMOD(LON1+G,TWOPI)
  200 CONTINUE
      RETURN
      END



      FUNCTION SYGN(Y)
! * * * * * * * * * * * * * * REAL FUNCTION SYGN * * * * * * * * * * * * * *
!     X=SYGN(Y)
!
!     This FUNCTION Returns the Value of 0 If Y Is 0, -1.  If Y Is Less
!               Than 0 and a +1.  If Y Is Greater Than 0.
!
!     SUBROUTINEs and FUNCTIONs Used:  None
!
! *** COMMON BLOCKS Referenced: None
      IF (Y) 100,200,300
  100 SYGN = -1.0
      GOTO 999
  200 SYGN = 0.0
      GOTO 999
  300 SYGN = 1.0
  999 RETURN
      END

      SUBROUTINE GCRAZ(TLAT,TLON,RLAT,RLON,RANGE,AZIM)
```

```
C
C     This routine was copied from "Quiet Time Lowest ObserVable
C     Fre-uency Model" by D. B. Sailors and W. K. Moision, NOSC
C     Tec  ical Report 1189, August 1987.
C
C     CALL GCRAZ(TLAT,TLON,RLAT,RLON,RANGE,AZIM)
C
C     This routine computes the great circle range and azimuth
C     between two points on the earth's surface.  TLAT and TLON
C     are the coordinates of point 1, RLAT and RLON are the
C     coordinates of point 2.  Both longitudes are west longitudes.
C     The output is RANGE, the distance between the two points
C     in radians, and AZIM, the azimuth from one point to the other
C     in radians.  This method assumes a spherical earth and
C     recognizes the degenerate cases of point 1 at the north
C     or south pole or points 1 and 2 coincident.  All coordinates
C     are in radians.
C
C     Subroutines and functions used:  none
C
C     common blocks:  none
C
C
C
      REAL    TLAT,TLON,RLAT,RLON
      DATA  PI/3.14159/,TWOPI/6.28318/,HALFPI/1.570796/
     &  ,DTR/0.017453/, RTD/57.295779/
C
C     TEST FOR DEGENERATE CASES
C
      IF (ABS(TLAT-HALFPI).GT. 1.0E-5) GO TO 100
C
C     POINT 1 IS AT THE NORTH POLE
C
      RANGE=HALFPI-RLAT
      AZIM=PI
      GO TO 140
  100 CONTINUE
      IF (ABS(TLAT+HALFPI) .GT. 1.0E-5) GO TO 120
C
C     POINT 1 IS AT THE SOUTH POLE
C
      RANGE=HALFPI+RLAT
      AZIM=0.0
      GO TO 140
  120 CONTINUE
      IF(ABS(TLAT-RLAT) .GT. 1.0E-5 .OR.
     &   ABS(TLON-RLON) .GT. 1.0E-5 ) GO TO 130
C
C     POINTS 1 AND 2 ARE COINCIDENT
C
      RANGE=1.0E-8
      AZIM=0.0
      GO TO 140
  130 CONTINUE
C
C     GENERAL CASE
C
      S1=SIN(TLAT)
      C1=COS(TLAT)
      S2=SIN(RLAT)
      CR=S1*S2+C1*COS(RLAT)*COS(TLON-RLON)
      CR=AMIN1(AMAX1(CR, -1.0),+1.0)
      RANGE=ACOS(CR)
      CA=(S2-S1*CR)/(C1*SIN(RANGE))
      CA=AMIN1(AMAX1(CA, -1.0), +1.0)
      AZIM=ACOS(CA)
      IF(SIN(TLON-RLON) .LT. 0.0) AZIM=TWOPI-AZIM
  140 CONTINUE
      RETURN
      END
```

```
C
C
C
       SUBROUTINE QLOF (CPNT,DTG,LOF)
CP
C      SUBROUTINE QLOF

C      CALL QLOF (CPNT,DTG,LUF)

C      THIS ROUTINE COMPUTES THE LUF FOR SOLAR QUIET CONDITIONS (WHEN
C      XRAY FLUX <1.85E-3).  THE REQUIRED INPUT IS CPNT WHICH IS  DE-
C      FINED IN SUBROUTINE  PATH  AND  ITIME WHICH  IS A SIX  ELEMENT
C      INTEGER ARRAY CONTAINING MONTH, DAY, HOUR, MINUTE, JULIAN DAY,
C      AND YEAR.  THIS ROUTINE RETURNS LUFS.

C      SUBROUTINES AND FUNCTIONS USED:  ABSORB

C      COMMON BLOCKS:  NONE
CZ
       DIMENSION CPNT(8)
       REAL            LOF, K, K3
       INTEGER          ITIME(6)
       CHARACTER*12 DTG
       D = CPNT(1)*6371.0
       LOF = 0.5
       CALL DHOUR(DTG,ITIME(6),ITIME(5),ITIME(1),ITIME(2),
      A   ITIME(3),FHOUR,REFHOUR)
C      CALCULATE LUF FOR PATH < 2000 KM
       IF (D .GT. 2000.) GOTO 200
       A = CPNT(1)/2.
       B = SIN(A)
       C = COS(A)
       CALL ABSORB (ITIME,CPNT(2),CPNT(3),AI,CHI)
       LOF = SQRT(AI/40./SQRT(1.-0.9784/(1.+((C-0.985)/B)**2)))
       GOTO 1000

C      CALCULATE LUF FOR PATH < 3300 KM
  200  IF (D .GT. 3300.) GOTO 400
       K = (4.0+1.875E-3*D)*0.045
       CALL ABSORB (ITIME,CPNT(2),CPNT(3),AI,CHI)
       LOF = K*SQRT(AI)
       GOTO 1000

C      CALCULATE LUF FOR PATH > 3300 KM
  400  K = (7.5+0.001*D)*0.045

C      CALCULATE LUF FOR PATH > 6600 KM
       IF (D .LE. 6600.) GOTO 500
       K3 = 1-0.3768*(CPNT(1)-1.0361)
       K = K3*K
  500  CALL ABSORB (ITIME,CPNT(2),CPNT(3),AB1,CHI)
       CALL ABSORB (ITIME,CPNT(4),CPNT(5),AB2,CHI)
       CALL ABSORB (ITIME,CPNT(6),CPNT(7),AB3,CHI)
       LOF = K*SQRT((AB1*2.0+AB2+AB3)/4.0)
 1000  CONTINUE
       LOF = AMIN1(AMAX1(LOF,0.5),50.0)
       RETURN
       END

       SUBROUTINE ABSORB (ITIME, LAT, LON, AI, CHI)
CP
C      SUBROUTINE ABSORB

C      THIS ROUTINE  COMPUTES THE  IONOSPHERIC  ABSORPTION  INDEX AT
C      POSITION LAT AND LON AT ITIME.  ITIME IS A  SIX ELEMENT INTE-
C      GER ARRAY FOR MONTH, DAY, HOUR, MINUTE, JULIAN DAY, AND YEAR.
C      LAT AND LON ARE IN RADIANS.  THE RESULTS OF THIS ROUTINE  ARE
C      STORED IN  AI (ABSORPTION INDEX)  AND CHI (SOLAR ZENITH ANGLE
C      IN RADIANS).  THIS  ROUTINE  WAS  WRITTEN  BY  DR. PAUL  ARGO
C      (NOSC).  LAT  AND  LON ARE  IN  RADIANS  AND  WEST LONGITUDES.
```

```
C       SUBROUTINES AND FUNCTIONS USED:  CH

C       COMMON BLOCKS:       PIDEGS
C                            SUN
CZ
        REAL            LAT, LON, M, N, LAD
        INTEGER         ITIME(6)

C        COMMON          /SUN/     SLAT,SLON
        CALL SUBSOL(ITIME,SLAT,SLON)
C         INCLUDE  PIDEGS.INC
        RTD=57.29577951
C       COMPUTE SOLAR ZENITH ANGLE
        W = 1.0
        CHI = SIN(LAT)*SIN(SLAT)+COS(LAT)*COS(SLAT)*COS(LON-SLON)
        CHI = AMAX1(CHI,-1.)
        CHI = AMIN1(CHI, 1.)
        CHI = ACOS(CHI)
        CHINON = ABS(SLAT-LAT)
        IF (CHINON .LT. 1.57) GOTO 80
        AI = 1.0E-13
        RETURN
   80   LAD = ABS(LAT)*RTD

C       TEST IF HIGH LATITUDE WINTER CORRECTION IS NEEDED
        IF (LAD .LT. 30.0) GOTO 100
        IF ((ITIME(1) .EQ. 1 .OR. ITIME(1) .EQ. 12) .AND. LAT .GT. 0.0)
     *     GOTO 90
        IF ((ITIME(1) .EQ. 6 .OR. ITIME(1) .EQ.  7) .AND. LAT .LT. 0.0)
     *     GOTO 90
        GOTO 100
   90   W = 1.0+0.0275*(30.0-ABS(60.0-LAD))

C       COMPUTE ABOSRPTION INDEX BASED ON LATITUDE
  100   CONTINUE
        ALAT = ABS(LAT)
        IF (ALAT .LT. 0.45) N=1.4-ALAT*2.44
        IF (ALAT .GE. 0.45 .AND. ALAT .LT. 1.0875) N=0.3
        IF (ALAT .GE. 1.0875 .AND. ALAT .LT. 1.367)
     *      N=-(ALAT-1.0875)*1.07+0.3
        IF (ALAT .GE. 1.367) N=0
        CSXN = COS(CHINON)**N
        ABSP = 286.0*W*(1.0+0.5*ALAT)*CSXN
        IF (ABSP .LT. 1.0E-11) ABSP=1.0E-11
        IF (LAD .GT. 18.0) GOTO 201
        M = 0.5*(0.58+(LAD/18.0)*0.08)
        GOTO 300
  201   CONTINUE
        IF (LAD .GT. 24.0) GOTO 202
        M = 0.5*(0.66+0.22*(LAD-18.0)/6.0)
        GOTO 300
  202   CONTINUE
        M = 0.44

C       ADJUST ACCORDING TO SOLAR ZENITH ANGLE IF DAYLIT
  300   CONTINUE
        AI = ABSP*0.01
        IF (CHI .GT. 1.8) RETURN
C          TYPE*,' ABSP=',ABSP,' CHI=',CHI,' CH(921.0,CHI)=',
C       A     CH(921.0,CHI),' CH(921.0,CHINON)=',CH(921.0,CHINON),
C       B      ' M=',M
        AI = ABSP*(CH(921.0,CHI)/CH(921.0,CHINON))**(-2.0*M)
        RETURN
        END



        FUNCTION CH (X,Y)
CP
```

```
C       REAL FUNCTION CH

C       X = CH(X,Y)

C       THIS FUNCTION COMPUTES THE CHAPMAN'S GRAZING INCIDENCE INTEGRAL
C       WHERE X IS THE PARAMETER RELATED TO THE ATMOSPHERIC DENSITY AND
C       SCALE HEIGHT, AND Y IS THE SOLAR ZENEITH IN RADIANS.  THIS ROU-
C       TINE IS ACCURATE TO  0.1%  WHEN  X*(U-SIN(I))<10  OR  COS(Y)>0.

C       SUBROUTINES AND FUNCTIONS USED:  CHPINT

C       COMMON BLOCKS:  CHPMN
CZ
        COMMON          /CHPMN/    XC, G, YC

        XC = X
        YC = Y
        CY = COS(Y)
        CYL = CY - 0.0174533
        IF (350.*Y .GT. X*CYL**4) GOTO 10
        CH = 1./CY
        RETURN
   10   G = X*SIN(Y)/(X+ALOG(X)+20.0)
        G = ATAN(G/SQRT(1.0-G*G))
        G = (G-Y)/20.0
        IF (CYL .LT. 0.0) GOTO 30
        IF (X*CYL .LT. 40.*Y)GOTO 20
        CH = -X*SIN(Y)*G*(.1464466*CHPINT(3.414214)
     *                 + .8535534*CHPINT(.5857864))
        RETURN
   20   CH = -X*SIN(Y)*G*(.5392947E-3*CHPINT(9.395071)
     *                   + .03888791*CHPINT(4.536620)
     *                   + .3574187*CHPINT(1.745761)
     *                   + .6031541*CHPINT(.3225477))
        RETURN
   30   CH = -X*SIN(Y)*G*(.4249314E-6*CHPINT(16.27926)
     *                   + .2825923E-4*CHPINT(11.84379)
     *                   + .7530084E-3*CHPINT(8.330153)
     *                   + .009501517*CHPINT(5.552496)
     *                   + .06208746*CHPINT(3.401434)
     *                   + .2180683*CHPINT(1.808343)
     *                   + .4011199*CHPINT(.7294545)
     *                   + .3084411*CHPINT(.1377935))
        RETURN
        END


        FUNCTION CHPINT (Z)
CP
C       REAL FUNCTION CHPINT
C       X = CHPINT(Z)

C       THIS ROUTINE COMPUTES THE INTEGRAND OF THE CHAPMAN INTEGRAL.

C       SUBROUTINES AND FUNCTIONS USED:  NONE

C       COMMON BLOCKS:  CHPMN
CZ
        COMMON          /CHPMN/    X, G, Y

        Q = Z*G
        U = SIN(Q+Y)
        Q = Q/2.0
        CHPINT = EXP(2.0*X*SIN(Q)*COS(Y+Q)/U+Z)/U/U
        RETURN
        END
```

```
      SUBROUTINE SUBSOL (ITIME,SLAT,SLON)
CP
C     SUBROUTINE SUBSOL
C     CALL SUBSOL(ITIME)
C     THIS ROUTINE COMPUTES THE COORDINATES OF THE SUB-SOLAR POINT
C     FOR A GIVEN TIME.  ITIME IS A SIX ELEMENT INTEGER ARRAY CON-
C     TAINING THE MONTH, DAY, HOUR, MINUTE, JULIAN DAY, AND  YEAR.
C     THE  COORDINATES  COMPUTED  ARE STORED IN  SLAT AND  SLON IN
C                              COMMON SUN.

C     SUBROUTINES AND FUNCTIONS USED:  NONE
C     COMMON BLOCKS:  SUN
CZ
      INTEGER          ITIME(6), YEAR
C      COMMON            /SUN/     SLAT, SLON
      DATA  A0/0.3798/,A1/-23.0009/
      DATA  A2,A3,A4,A5,A6/-0.3802,-0.1550,-0.0076,-0.0025,-0.0004/
      DATA  B1,B2,B3,B4,B5/ 3.5354, 0.0302, 0.0728, 0.0032, 0.0020/
      DATA  C1,C2,C3,C4,C5/ 0.5965,-2.9502,-0.0653,-0.1248,-0.0103/
      DATA  D1,D2,D3,D4,D5/-7.3435,-9.4847,-0.3083,-0.1747,-0.0159/
      DATA  ONE,TWO/1.0,2.0/
      DATA  TWOPI/6.28319/,DTR/0.0174533/

      IHOUR = ITIME(3)
      MINUTE = ITIME(4)
      JDAY = ITIME(5)
      YEAR = ITIME(6)
      DAY = JDAY+IHOUR/24.0+MINUTE/1440.0

      K = MOD(YEAR,4)
      DATE = 365.0*FLOAT(K)+0.0078*(FLOAT(YEAR)-68.0)
      IF (K .NE. 0) DATE=DATE+1.0
      DATE = DATE+DAY
      X = DATE*TWOPI/365.25
      SX = SIN(X)
      CX = COS(X)
      TWOCX = TWO*CX
C     RECURSIVE CALCULATION FOR COS(N*X) AND SIN (N*X),N=2,6
      C2X = TWOCX*CX-ONE
      S2X = TWOCX*SX
      C3X = TWOCX*C2X-CX
      S3X = TWOCX*S2X-SX
      C4X = TWOCX*C3X-C2X
      S4X = TWOCX*S3X-S2X
      C5X = TWOCX*C4X-C3X
      S5X = TWOCX*S4X-S3X
      C6X = TWOCX*C5X-C4X
C     FOURIER EXPANSION FOR SOLAR DECLINATION AND EQUATION OF TIME
      DEC = A0+A1*CX+A2*C2X+A3*C3X+A4*C4X+A5*C5X+A6*C6X
     *        +B1*SX+B2*S2X+B3*S3X+B4*S4X+B5*S5X
      EQNT = C1*CX+C2*C2X+C3*C3X+C4*C4X+C5*C5X
     *       +D1*SX+D2*S2X+D3*S3X+D4*S4X+D5*S5X
      GHA = IHOUR-(12.0-EQNT/60.0)+MINUTE/60.0
      SLAT = DEC*DTR
      SLON = 15.0*GHA*DTR
      IF (SLON .LT. 0.0) SLON=SLON+TWOPI
      RETURN
      END
```

TABLE 1
Sunspot Number of the 133-Month Mean Sunspot Cycle Used in Subroutine MEANSSN

MONTH	SSN	MONTH	SSN	MONTH	SSN
0	5.0	45	104.5	90	45.0
1	5.0	46	104.5	91	44.0
2	5.5	47	105.0	92	43.0
3	6.0	48	105.0	93	41.5
4	6.5	49	105.0	94	40.0
5	7.0	50	104.5	95	38.0
6	8.0	51	104.5	96	37.0
7	9.0	52	104.0	97	35.5
8	10.0	53	104.0	98	34.0
9	11.0	54	103.5	99	33.0
10	12.0	55	103.0	100	32.0
11	14.0	56	102.0	101	30.5
12	16.0	57	101.5	102	29.5
13	20.0	58	100.5	103	28.5
14	21.0	59	99.5	104	27.0
15	24.0	60	98.5	105	26.0
16	26.5	61	98.0	106	25.0
17	31.0	62	97.0	107	24.0
18	34.0	63	96.0	108	23.5
19	38.0	64	95.0	109	22.5
20	42.0	65	93.0	110	22.0
21	45.0	66	91.5	111	20.5
22	48.0	67	89.0	112	20.0
23	53.5	68	87.0	113	19.0
24	57.0	69	85.0	114	18.0
25	60.0	70	83.0	115	17.5
26	63.0	71	81.0	116	17.0
27	67.5	72	78.5	117	16.0
28	70.0	73	76.0	118	15.0
29	74.5	74	74.0	119	14.0
30	78.0	75	70.5	120	13.5
31	80.0	76	68.5	121	13.0
32	83.5	77	66.5	122	12.5
33	85.5	78	65.0	123	12.0
34	88.0	79	62.0	124	11.0
35	92.0	80	60.0	125	11.0
36	94.0	81	59.0	126	10.0
37	96.5	82	56.5	127	10.0
38	97.5	83	55.0	128	(9.8)
39	100.0	84	53.0	129	(9.6)
40	101.0	85	52.0	130	(9.4)
41	102.0	86	51.0	131	(9.2)
42	103.0	87	49.0	132	(9.0)
43	103.5	89	47.5		
44	104.0	88	48.0		

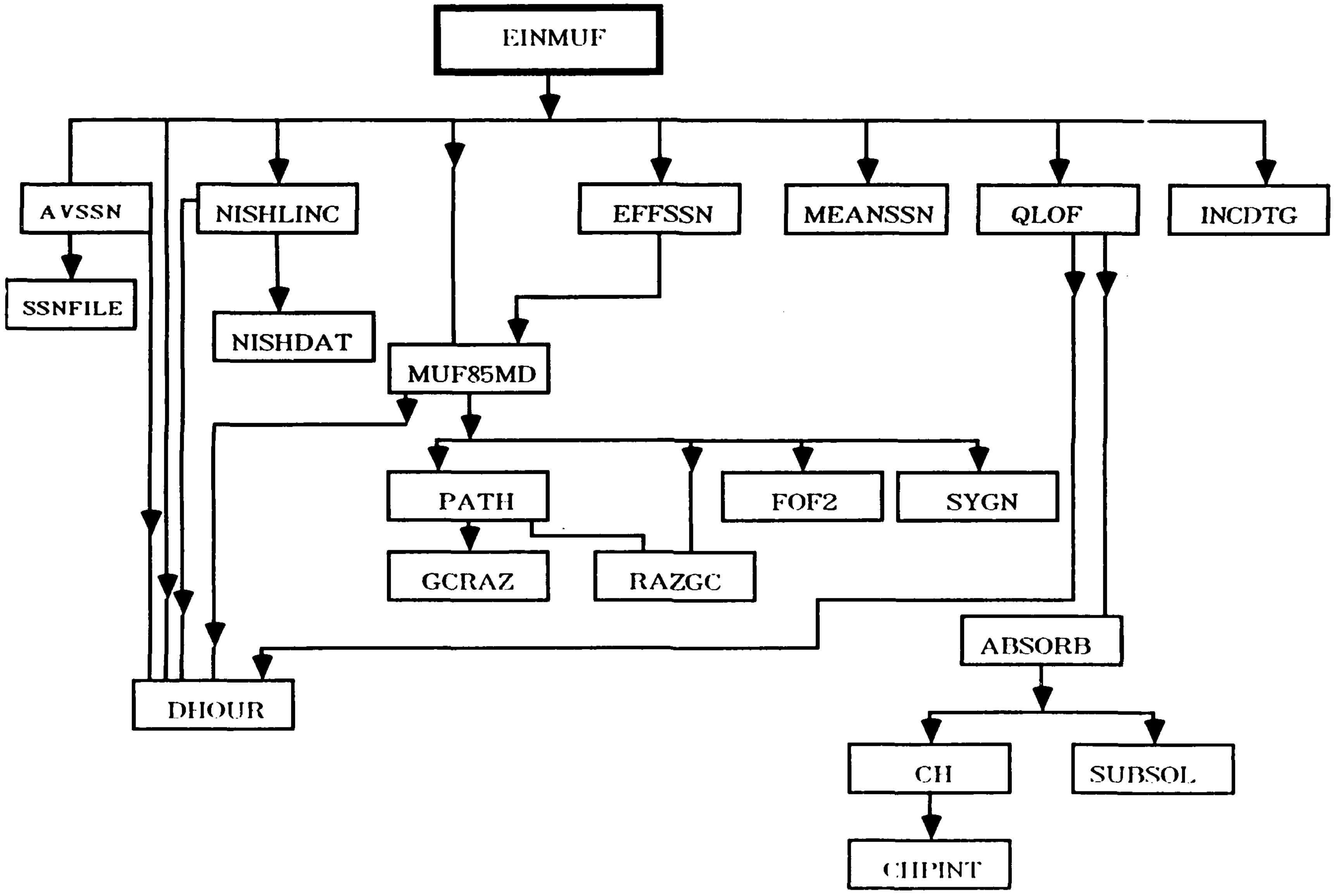

Figure 1. Paths connecting EINMUF program elements.

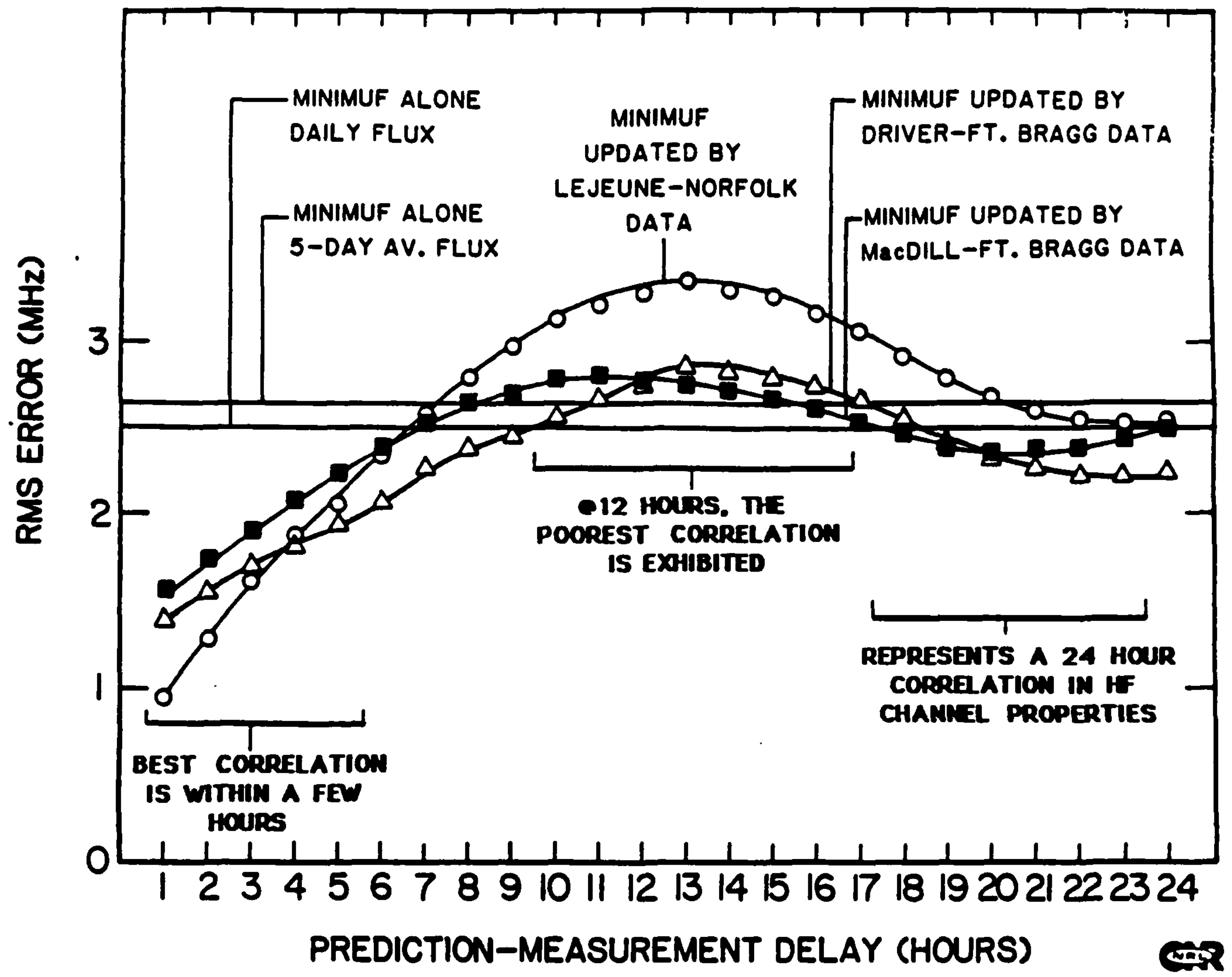

Figure 2. Rms error between actual MOFs on the Lejeune-Norfolk path and MOFs predicted by a variety of techniques, as a function of time delay between prediction and measurement. Data include approximately 335 hourly predictions for each value of time delay. This figure taken from Daehler (1984).

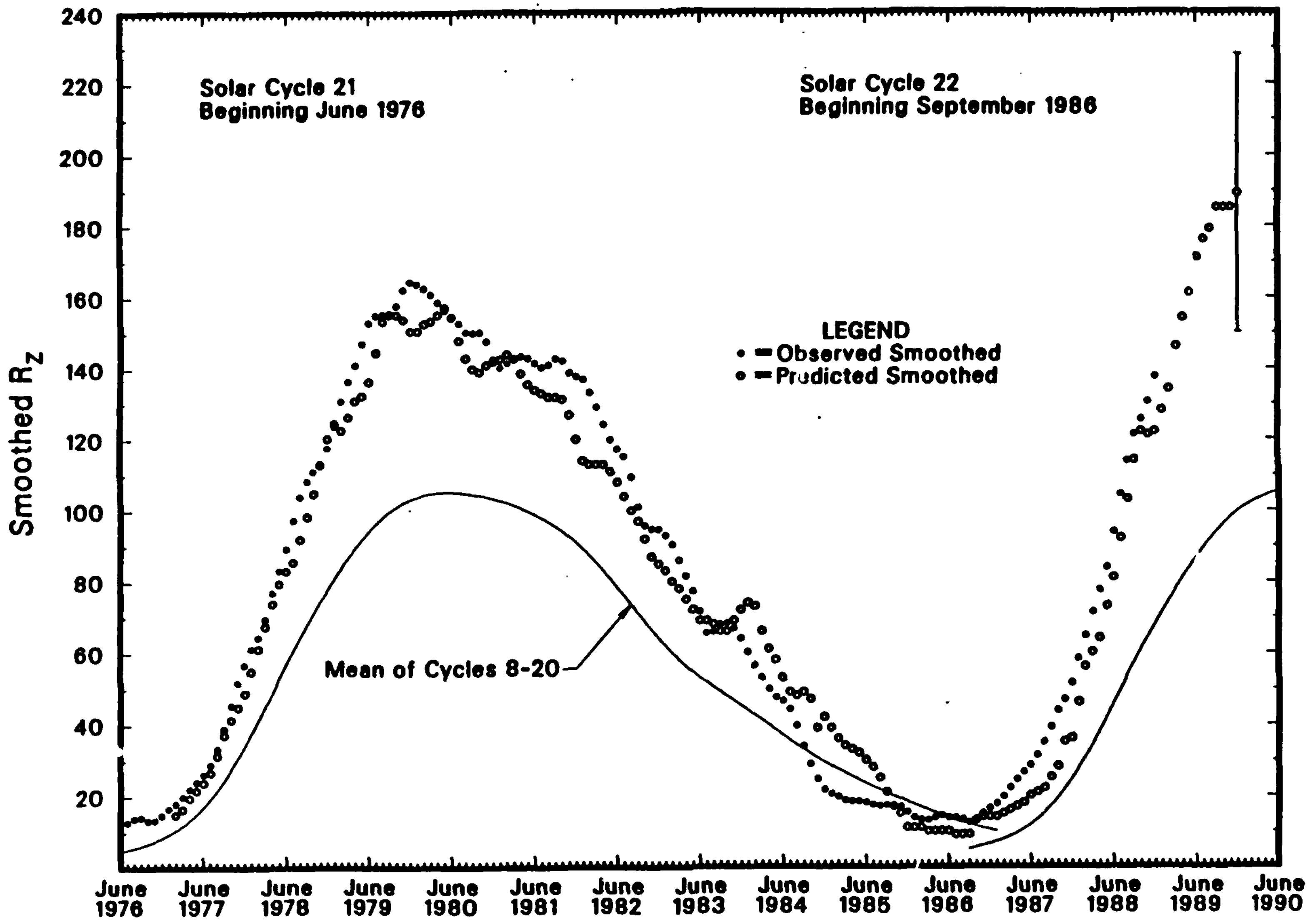

Figure 3. Observed and predicted monthly sunspot numbers. The solid line represents the mean sunspot number measured for solar cycles 8-20. Observed and predicted sunspot numbers for cycle 21 and the beginning of cycle 22 (the current cycle, starting September 1986) are represented by solid and open circles, respectively. (From Solar-Geophysical Data Prompt Reports, Nbr. 539, Part I, Page 17, July 1989.)

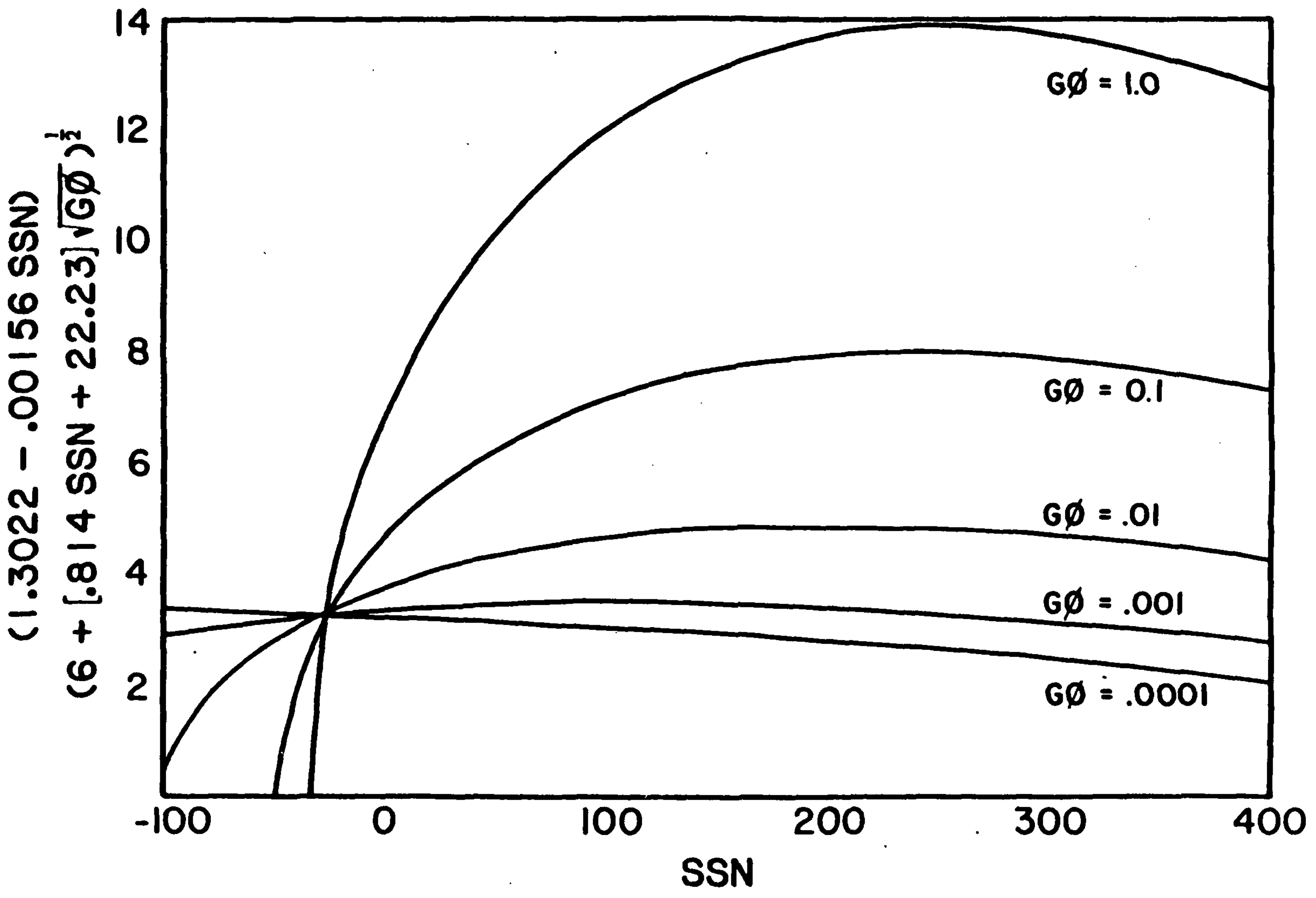

Figure 4. The function G(SSN,GØ) = (1.3022 - .00156SSN)SQRT(6 + [.814SSN + 22.23]SQRT(GØ)) from MINIMUF85. The MINIMUF85 MUF for a given path and time is proportional to this quantity, which contains the sunspot dependence. The variable GØ is the effective Zenith angle.